TITAN USERS GUIDE
INSTALLING TITAN

Mike Dixon
Research Applications Laboratory
National Center for Atmospheric Research
Boulder Colorado USA

Access to TITAN software - intellectual property issues

The intellectual property (IP) rights to TITAN are held by the University Corporation for Atmospheric Research (UCAR).

For research and small non-profit organizations, access to use TITAN is granted free of charge by UCAR, provided the user accepts the terms of a web-based UCAR license agreement which indemnifies UCAR against liability.

This web license agreement can be found at:

`www.rap.ucar.edu/projects/titan`

Larger non-profit organizations and government agencies may also use TITAN free of charge. However, such organizations are required to sign a formal license agreement with UCAR indemnifying UCAR against liability. For-profit organizations are also required to sign a formal license agreement with UCAR, and in some cases a license fee may apply.

TITAN is provided as-is, with no warranty for any purpose.

LINUX platform specifications

LINUX is the only fully-supported platform for running TITAN. Other operating systems have been supported in the past, and TITAN should still run on these, but the latest version of TITAN has only been extensively tested under LINUX.

The following are suggestions for the specifications for your LINUX platform, along with libraries which are needed for various applications.

Hardware specifications

- CPU - 3.0+ GHz. For real-time (and especially time-series work), fast quad-core processors are recommended.
- RAM - 2 to 8 GBytes

- Disk space: large disks are good for archiving data. At a minimum, you should have a 200 Gbyte disk for a real-time system. For data archival, 500 GByte to 1 TByte is good.
- Graphics card: 256+ MByte memory for 1600x1200 screen resolution.
- Check for LINUX compatibility. For specific hardware devices, it is worth checking, via Google searches, that a good driver is available. This should preferably be a Debian-supported driver with source code so you can compile it if necessary.

Packages

- C++ compiler: gcc/g++ 4.2 or later.
- FORTRAN compiler: gfortran (for grib support).
- X11R6
- fftw3
- qt4 (for selected applications, not required for most apps)
- perl
- ImageMagik development libraries:
  `imlib2, gif, jpeg, png, tiff`
  (NOTE: these ImageMagik libs are only needed for the CIDD display application. If you do not have imlib2, you can use the apps/titan/src/CIDD_titan version, which by default does not need imlib. The main difference is in the web content generation capability. If you do not need content generation, use either version.)
- `bzip2` library for `nexrad2dsr`, which needs `libbzip2` to unzip the incoming files.

Java-related packages

If you need access to the Java applications – SysView and Jadeite, you will also need:

- java 1.6 run-time environment (jre)

And for compiling Java apps:

- java 1.6 development kit (jdk)
- ant

- ant-contribs

TITAN user account

TITAN can be run under any account.

It is **not advisable** to run it as **root**.

For real-time systems, TITAN is normally run under the **titan5** user account. This documentation will assume this is the account name.

User account login shell

The user account should use either **csh** or **tcsh** as the login shell. On LINUX these are identical.

Where to get TITAN

The main TITAN web site is:

www.rap.ucar.edu/projects/titan

Downloading TITAN

For downloading, go to the TITAN web site, select the **Download** tab, followed by **Perform download**.

Fill out the details in the license agreement, and click **I agree**. You will then be transferred to the download page.

The latest TITAN distribution source will appear as a gzipped tar file. For example, the file:

```
titan5.20090825.src.tgz
```

is the source code distribution from 2009/08/25.

Included on the download page are tar files which may be useful for building 3rd-party libraries on which parts of TITAN depend, such as:

- fftw3 – FFT library (required)
- NetCDF and udunits2 from Unidata (only needed for NetCDF apps)
- HDF5 (only needed for NetCDF apps)

Note: you can either download these tar files, or get the latest from the relevant sites on the web.

Make a 'rap' subdirectory (~/rap) under the titan5 user home directory.

(NOTE: '~' in UNIX is short-hand for the home directory of the active account. So ~/rap is the same as /home/titan5/rap, if the user is titan5.)

Download the TITAN tar file (see above) and put the tar file there. For example:

```
cd
mkdir ~/rap
cp titan5.20090825.src.tgz ~/rap
cd ~/rap
tar xvfz titan5.20090825.src.tgz
```

After this step, the rap directory should contain something like the following:

`apps/` (C++ applications source)
`build_titan` (main build script)
`build_titan_apps` (build applications only)
`build_titan_libs` (build libraries only)
`check_titan_apps` (check the installed apps)
`check_titan_libs` (check the installed libs)
`clean_titan_build` (clean the build, ready for another)
`distribs/` (distribution files)
`doc/` (documentation)
`incs/` (system include files)
`install_cshrc` (boot-strap script for installing .cshrc file)
`java/` (java source code)
`libs/` (C++ library source code)
`LICENSE` (license details)
`make_include/` (include files for make system)
`projects/` (configuration files for TITAN projects, templates etc)
`README_BUILD` (README for building libs and apps)
`README_INSTALL` (README for installing project configuration files and scripts)
`README_LINUX_SETUP` (README on setting up LINUX)
`RELEASE_NOTES/` (release notes, by date)
`set_build_env` (script for setting the build environment)
`TERMS_OF_USE` (more license info)
`WARRANTY_DISCLAIMER` (more license info)

The various scripts you need to build and install the system have been copied to this top level.

Also, the important `README` files and `RELEASE_NOTES` have been copied here. You will need to refer to these files to find out how to build and install the release you have downloaded.

The `RELEASE_NOTES` directory contains files with notes about recent releases. Releases are identified by date rather than by number, because TITAN is a collection of a large number of applications each with individual version numbers.

Steps in setting up TITAN

The following steps are required to set TITAN up on a LINUX host:

1. install LINUX and packages (see above)
2. download TITAN distribution (see above)
3. build TITAN libraries and applications
4. install TITAN project configuration and scripts

Step 3, the build process, installs the libraries, applications and common scripts in subdirectories of the ~/rap directory.

Step 4, the install process, either copies TITAN configuration files and scripts into the project directory, or makes a link from the home directory into the project section of the distribution. New users will use the standard templates in the TITAN distribution for this step, as a way of exploring the system and learning how it operates. However, since projects vary a great deal, step 4 for most users will involve installing their own configuration rather than using one of the standard templates.

Building and installing TITAN

The following README files contain the details needed to set up LINUX, build the TITAN applications and install TITAN:

- `README_LINUX_SETUP`
- `README_BUILD`
- `README_INSTALL`

Performing the build

Follow the procedure in `README_BUILD.`

This will build the code optimized for performance.

Boot-strapping the build environment

When you start with a clean account, you need to set up an environment for the build and installation steps. This is itself a step-by-step procedure, as follows:

- `cd ~/rap`
- run `install_cshrc`, which installs a default `~/.cshrc` file in the titan5 home

directory

- run `set_build_env`, which creates the file `~/.titan_build_env`. This file is then used by the .cshrc file for getting the build environment variables.
- run `source ~/.cshrc` to make the environment active

The 'bootstrap' is then complete, and the build may be performed.

Build environment

The build scripts:

```
build_titan
build_titan_libs
build_titan_apps
```

attempt to build the TITAN libraries and applications and install them in ~/rap. They make use of the build environment set up in the bootstrap step.

The environment variables set up by the bootstrap are stored in the file ~/.titan_build_env. The most important variables are:

`HOST_OS` – operating system type
`BUILD_DIR` – top level directory for build, normally ~/rap
`RAP_MAKE_INC_DIR` – directory for make include files
`RAP_INC_DIR` – directory for installed include files
`RAP_LIB_DIR` – directory for installed library files
`RAP_BIN_DIR` – directory for installed binary files

The `HOST_OS` should be one of the following:

OS	**HOST_OS**
32-bit LINUX	LINUX
64-bit LINUX	LINUX_64
64-bit LINUX, 32-bit mode	LINUX_64_32MODE
32-bit Solaris	SUNOS5
64-bit Solaris	SUNOS5_64
MAC-OS	DARWIN

Performing a project installation

Follow the procedure in `README_INSTALL`

Testing an installed project

The project templates include a test environment to allow you to run a typical project using simulated data.

Details for doing this are included in `README_INSTALL`.

Troubleshooting the build process

Below is detailed information about the build environment and utilities. This information may be helpful in troubleshooting problems with the build.

How 'make' works

The UNIX `make` utility is used to build the TITAN system. `make` uses the configuration files '`makefile`' or '`Makefile`' to control the build. Note: if both the upper case `Makefile` and lower case `makefile` are present, the lower case variant is used.

The Makefiles in TITAN use the 'include' feature of `make` to keep individual Makefiles simple and easy to read and to provide the flexibility to handle different operating systems etc.

The `make_include` directory, as it is known, is found at:

```
~/rap/make_include
```

As an example of a Makefile, let us take a look at:

```
~/rap/apps/ingest/src/Ltg2Spdb
```

The following is a listing of the Makefile:

```
include $(RAP_MAKE_INC_DIR)/rap_make_macros
TARGET_FILE = Ltg2Spdb
LOC_INCLUDES =
LOC_CPPC_CFLAGS =
LOC_LDFLAGS =

LOC_LIBS = -lSpdb -ldsserver -ldidss -lrapformats \
          -ltoolsa -ldataport -ltdrp -lpthread

HDRS = \
```

```
                $(PARAMS_HH) \
                Args.hh \
                Ltg2Spdb.hh

CPPC_SRCS = \
                $(PARAMS_CC) \
                Args.cc \
                Main.cc \
                Ltg2Spdb.cc

#
# tdrp macros
#
include $(RAP_MAKE_INC_DIR)/rap_make_tdrp_macros
#
# standard targets
#
include $(RAP_MAKE_INC_DIR)/rap_make_c++_targets
#
# tdrp targets
#
include $(RAP_MAKE_INC_DIR)/rap_make_tdrp_c++_targets
```

The complexity of the Makefile is hidden by the various includes. Note how we use the environment variable $`RAP_MAKE_INC_DIR` to point to the include directory.

For most purposes, the user of the Makefile need only be concerned with the list of source files and the name of the application or library module.

Make targets

A make ‘target’ is the action you want make to take. The various make targets are defined in the `make_include` files. Targets may be chained on the command line - make will execute them in order.

If you just type ‘`make`’, you will invoke the default target, which is normally ‘`all`’. This is set up to take the most likely action, which in most cases is to compile the library or application with the debug flag set on. For specific applications, the default action is to compile optimized.

For use in the field, TITAN should be built optimized.

The most common targets used are:

- `opt`: compile optimized
- `install`: install the built library or application

- `install_include`: install the include files for a library. This is generally done for all libraries at the start of the build, so that the include files are available for all libraries and applications.
- `clean_all`: clean out all of the object files and executables. May be done after an install to free up disk space.
- `clean_bin`: removes just the executable file. '`make clean_bin all`' forces a relink.
- `clean_lib`: removes the '`.a`' file. 'make clean_lib all' creates a fresh '`.a`' file.

Recursive makes

The Makefile system is recursive, meaning that you can execute make commands from various parts of the tree.

The following examples show how you can execute make from a variety of locations:

- `cd ~/rap/incs; make install:` install the system wide include files
- `cd ~/rap/libs; make install_include opt install`: for all libraries, install include files, performed optimized compile, install libraries
- `cd ~/rap/libs/toolsa; make clean opt install`: performed clean optimized compile and install toolsa
- `cd ~/rap/libs/toolsa/src/Path; make clean opt`: perform clean optimized compile on just the Path module of toolsa.
- `cd ~/rap/apps; make opt install`: optimized compile and install for all applications.
- `cd ~/rap/apps/ingest: make opt install`: optimized compile and install of all ingest applications.
- `cd ~/rap/apps/titan/src/Rview; make opt install`: optimized compile and install for Rview app only.

Directories for binaries, libraries and code

After building, the installed directory structure should be as follows:

`~/rap/bin` (binaries and scripts)
`~/rap/lib` (libraries)
`~/rap/lib/perl5` (perl modules)
`~/rap/include` (include files)
`~/rap/make_include` (include files for make system)

If you are building on a 64-bit system, you will also see:

`~/rap/bin64` (binaries and scripts)
`~/rap/lib64` (libraries)

If you are building on a 64-bit system in 32-bit mode, you will also see:

`~/rap/bin32` (binaries and scripts)
`~/rap/lib32` (libraries)

Building individual libraries or applications

You may wish to build components of the system individually, either because they did not build properly or because they were not include as part of the standard build. The second reason applies to applications which need some extra library, such as netCdf, which is not distributed with TITAN. These applications must be built separately after the necessary libraries are installed.

Then, you should be able to build any library or application individually.

For libraries, we use toolsa as an example:

```
cd ~/rap/libs/toolsa/src
make opt install
```

For applications, we use Titan as an example:

```
cd ~/rap/apps/titan/src/Titan
make opt install
```