Linux Core Installation

Guide

This article is part of the Installation Guide. You can read it alone or click on the previous link to easily move between the steps.
<< Step 1: Requirements Step 3: Server Setup >>

Content

Getting started

This how-to will attempt to help with TrinityCore, and also show how this can be done in a way that also shows you the basics of how linux compilations works.

> Most of this how-to is based on the use of a Debian based distribution, though we'll try to inform as best as we can when something differs totally.

> **Hint**
> (Master only) If you plan to run compile and run TrinityCore on a Linux machine and the World of Warcraft client on a separate Windows PC, you will also need to compile it on the Window PC, so that you have a Windows "connection_patcher.exe" binary, which needs to be run on the machine where the client will run.

Advices:

- Read your distributions' documentation on how to install packages, and also have at least knowledge on how it works with regards to adding users.*
- Run/install TrinityCore on a dedicated machine, or a machine that you know you have full control over.
- Do NOT install the software on a shared server solution or any server where other users may have access or might require resources to be available at all times.

Your server may be abruptly killed by an angry administrator or system staff for overuse of system resources.

Creating a user to work with

Start with logging in to your Linux-machine and create an account for the server itself - on most recent distributions this can easily be done with the following command :

```
sudo adduser <username>
```

Note : Change <username> into the preferred username of your server-account - we will as far as possible avoid using specific usernames in this how-to.

Sample usernames found in various parts of this guide: ***wow*** *,* ***trinity*** *(- select a logical name that makes sense to you when creating the user -).*

```
sudo su - <username>
```

Note : Change your current user to <username> so everything will run and compile with the user you just have created.

Required software

See Requirements

Building the server itself

Getting the source code

3.3.5 (WOTLK CLIENT)

```
cd ~/
git clone -b 3.3.5 git://github.com/TrinityCore/TrinityCore.git
```

This will clone 3.3.5a branch, this is the **RECOMMENDED** branch for starters.

MASTER (LEGION CLIENT)

```
cd ~/
git clone -b master git://github.com/TrinityCore/TrinityCore.git
```

This will clone **master** branch, note that this is **NOT** the recommended branch for starters.

The directory **TrinityCore** will be created automatically and all the source files will be stored in there.

> FreeBSD users will need to apply the patch located here for g3d to compile properly, before doing anything else.

Compiling the source code

Creating the build-directory

To avoid issues with updates and colliding source builds, we create a specific build-directory, so we avoid any possible issues due to that (if any might occur)

```
cd TrinityCore
mkdir build
cd build
```

Configuring for compiling

To configure the core, we use space-separated parameters attached to the configuration-tool (cmake) - **do read the entire section before even starting on the configuration-part**.
This is for your own good, and you HAVE been warned. A full example will also be shown underneath the explanations.

```
cmake ../ [additional parameters]
```

Parameter explanations (advanced users) (expand)

Parameter explanations

path to your OpenSSL library - **use if you have OpenSSL installed system wide with a different version to 1.0.x**:

```
-DOPENSSL_LIBRARIES=<path to OpenSSL library>
```

path to your OpenSSL includes directory - **use if you have OpenSSL installed system wide with a different version to 1.0.x**:

```
-DOPENSSL_INCLUDE_DIR=<path to OpenSSL includes>
```

Parameter	Values	Usage	Default
DSCRIPTS DSCRIPTS_COMMANDS DSCRIPTS_CUSTOM DSCRIPTS_EASTERNKINGDOMS DSCRIPTS_EVENTS DSCRIPTS_KALIMDOR DSCRIPTS_NORTHREDN DSCRIPTS_OUTDOORPVP DSCRIPTS_OUTLAND DSCRIPTS_PET DSCRIPTS_SPELLS DSCRIPTS_WORLD	• **"none"**: Disables all scripts • **"static"**: - Builds all scripts statically. (this is the old -DSCRIPTS=1 option). • **"dynamic"**: - Builds all scripts dynamically (Experimental feature). • **"minimal-static"**: Builds Commands and Spells statically, disables other scripts (this is the old -DSCRIPTS=0 option). • **"minimal-dynamic"**: Builds Commands and Spells dynamically, disables other scripts. **DSCRIPTS modules** • **"default"**: Inherit DSCRIPTS value. • **"disabled"**: Disables the building of the given module. • **"static"**: Compiles the given module as a static library. • **"dynamic"**: Compiles the given module as a dynamic library.	-DSCRIPTS="dynamic" -DSCRIPTS_COMMANDS="default" -DSCRIPTS_OUTDOORPVP="disabled" -DSCRIPTS_PET="static"	"static"
DTOOLS	• **1**: Builds map/vmap/mmap etractor/assembler and connection_patcher (6.x) • **0**: Disables the building of tools.	-DTOOLS=0	1
DSERVERS	• **1**: Builds authserver and worldserver. • **0**: Disables the building of servers.	-DSERVERS=1	1
DCMAKE_INSTALL_PREFIX	Path to the directory where TrinityCore will be installed. Required for script hotswapping.	-DCMAKE_INSTALL_PREFIX=/home/trinitycore/bin	/usr/local
DWITH_WARNINGS	• **1**: Show all warning during compile. (**Advanced users only**) • **0**: Disable most warnings during compile.	-DWITH_WARNINGS=1	0
DUSE_COREPCH	• **1**: Use precompiled headers when compiling servers. • **0**: Disables precompiled headers during servers compilation. (**Advanced users only**)	-DUSE_COREPCH=0	1
DUSE_SCRIPTPCH	• **1**: Use precompiled headers when compiling scripts. • **0**: Disables precompiled headers during servers compilation. (**Advanced users only**)	-DUSE_SCRIPTPCH=0	1
DCONF_DIR	Sets configuration directory. (**Advanced users only**)	-DCONF_DIR=/home/trinitycore/conf	/usr/local/etc

DLIBSDIR	Sets libraries directory. (**Advanced users only)**	-DLIBSDIR=/lib	/usr/local/lib
DCMAKE_C_FLAGS	Set C_FLAGS for compilation. (**Advanced users only)**	-DCMAKE_C_FLAGS="-msse3 -O3"	
DCMAKE_CXX_FLAGS	Set CXX_FLAGS for compilation. (**Advanced users only**)	-DCMAKE_CXX_FLAGS="-std=c++11 -O0"	

The above parameters when combined into a full example :

```
By default this is the only row you will need to run to setup your
install:
cmake ../ -DCMAKE_INSTALL_PREFIX=/home/<username>/server

Another Examples Below:
cmake ../ -DCMAKE_INSTALL_PREFIX=/home/wow/server -DTOOLS=0
cmake ../ -DCMAKE_INSTALL_PREFIX=/home/$USER/server -DTOOLS=0
-DWITH_WARNINGS=1
```

The 1st row builds the core with the tools, set installation base directory to **/home/<username>/server**.

The 2nd row builds the core without the tools, set installation base directory to **/home/wow/server**.

The 3nd row builds the core without the tools, set installation base directory to **/home/$user/server and enables warnings.**

NOTE: If you see
"-- Performing Test boost_filesystem_copy_links_without_NO_SCOPED_ENUM - Failed" IGNORE, it's a warning.

Note that you WILL have to configure the server well if you ever want to use the RA-access functionality.

Building the core

After configuring and checking that everything is in order (read cmakes output), you can build Trinity (this will take some time unless you are on a rather fast machine)

```
make
make install
```

If you have multiple CPU cores, you can enable the use of those during compile :

```
make -j <number of cores>
make install
```

Alternatively:

```
make -j $(nproc) install
```

After compiling and installing, you will find your core binaries in **/home/<username>/server/bin**, and the standard configuration files in the **/home/<username>/server/etc** folder.
(As usual, replace <username> with the username you created earlier). Now you can continue reading on and learn how how to update the source tree.

Keeping the code up to date

TrinityCore developers are always at work fixing and adding new features to the core. You can always check them here. To update the core files, do the following :

```
cd ~/TrinityCore/
# For 3.3.5 Branch
git pull origin 3.3.5

# For master Branch
git pull origin master
```

Now return to the compilation-section again, and repeat the instructions there.

Installing MySQL Server

When configuring MySQL make sure you remember the password you set for the default root account and that you enabled both MyISAM and InnoDB engines.

SPECIAL NOTES! you don't need to read this unless you want to do custom instalations.

Things to notice :

> -DWITH_COREDEBUG=0 **not required, this flag is only for core developers as its default is : 0 it may cause crashes if using on production environments if you want to compile core on debug mode you need to use** -DCMAKE_BUILD_TYPE=Debug

The new method for custom SSL-libraries are:

> -DOPENSSL_LIBRARIES=<path to OpenSSL libraries directory>
> -DOPENSSL_INCLUDE_DIR=<path to OpenSSL br /includes directory>

The paths for installation can be done without any other parameters but this :

> -DCMAKE_INSTALL_PREFIX=/path/to/where/you/want/core/to/be/installed

It will create the following structure:

> <path>/bin/ **- binaries will be placed here**
> <path>/etc/ **- config files will be placed here**

Also, compile has been tested on Debian 9 x32/x64, Ubuntu 17.10/18.04 x64 - all without problems IF YOU DO NOT MESS AROUND ON YOUR OWN!
We don't recommend to mix deps on older linuxes and update your distribution to one modern linux (debian 9, ubuntu 18.04)

Please remember to rename the **worldserver.conf.dist** and **authserver.conf.dist** files in **worldserver.conf** and **authserver.conf** respectively, unless you want to keep the configuration files of a previously compiled version of the core.

Guide

This article is part of the Installation Guide. You can read it alone or click on the previous link to easily move between the steps.

<< Step 1: Requirements

Step 3: Server Setup >>