

---

# Alien Arena Developer Reference

## Table of Contents

1. About this Document .....	1
2. Subversion Repository .....	2
3. Visual C++ 2010 Build .....	2
3.1. Windows Subversion Client .....	3
3.2. Microsoft Visual C++ 2010 Express .....	3
3.3. Libraries .....	3
3.4. OpenAL 1.1 SDK .....	4
3.5. Directory Tree .....	4
3.6. The config.h file .....	6
4. Unix/Linux Build .....	7
4.1. Introduction .....	7
4.2. Program Changes in 7.50 .....	7
4.3. Subversion Repository .....	8
4.4. Maintainer's Guide .....	8
4.5. Packager's Guide .....	12
4.6. Configure options .....	14
4.7. GCC Compiler Options .....	15
4.8. config.h .....	16
4.9. Open Dynamics Engine (ODE) Integration .....	16
5. MinGW Build .....	16
6. Mac OS X / Darwin X11 Build .....	16
7. Developer's Not-Autotools Makefile .....	16
8. Desura .....	17
9. Static Analysis .....	17
10. Phoronix Test Suite .....	17

Copyright © 2011 COR Entertainment, LLC.

```
$Id: devref-en.asciidoc 3224 2012-10-06 06:13:22Z strat $
```

## 1. About this Document

### Warning

*This is an informal document and assumes general knowledge of programming and specific knowledge of Alien Arena development.*

This document is intended to be informal reference for Alien Arena development. Its reason for being is it organize general information, notes, and details about development tools, build procedures, packaging, etc. In 2010, the Windows build was converted to use Microsoft Visual Studio 2010 and the Unix/Linux version was converted to use GNU Autotools. Also, some work was done on MinGW and Mac OS X/Darwin X11 builds. A big reason for creating this document was to organize the important details about all that in one place.

The source for this document uses AsciiDoc. Briefly, AsciiDoc is a frontend for a variety of XSL document programs and runs on Linux. Linux distributions have it in their package managers. Running it on Windows apparently requires using Cygwin, unfortunately. In some ways, it is "DocBook-for-Dummies" and is relatively easy to edit.

The home page for AsciiDoc is <http://www.methods.co.nz/asciidoc/>.

The SourceForge repository is <http://sourceforge.net/projects/asciidoc/>.

In the *alienarena/docs* directory these are the commands used to generate output.

text

```
a2x --format=text --destination-dir=. docsrc/devref-en.asciidoc
```

html

```
a2x --format=xhtml --destination-dir=. --stylesheet=CSSPATH docsrc/devref-en.asciidoc
```

pdf

```
a2x --format=pdf --fop --destination-dir=. --stylesheet=CSSPATH docsrc/devref-en.asciidoc
```

Where *CSSPATH* is something like: *docbook-xsl.css*

Which references something like: */usr/share/asciidoc/stylesheets/docbook-xsl.css*

## 2. Subversion Repository

Alien Arena resources are stored in a Subversion (SVN) repository at <http://icculus.org>. To browse the repository, go to <http://svn.icculus.org/alienarena/trunk>. Use a Subversion client program for downloading. Read access to the repository is not restricted. Irritant administers commit access.

### Caution

The SVN versions of the *.sln* and *.vcxproj* files are official. Do not commit these without clearance from Irritant. Doing so could cause confusion and consternation.

## 3. Visual C++ 2010 Build

**Steps for building Release version with Visual C:**

- Install TortoiseSVN.
- Checkout Alien Arena from SVN.
- Install and register Microsoft Visual C++ Express 2010.
- Install the OpenAL SDK.
- Install the libraries from the *.zip* files.
- Install the libcurl patch.
- Run *premake* to generate *config.h* for libode.

- Open the *.sln* file in Visual Studio.
- Select the Release build and run it.
- Copy *VS2010\bin\Release\alienarena.exe* to the top directory.

### Note

For the libode *config.h* (and maybe some other files), an alternative is to get a copy from someone who has a working installation.

## 3.1. Windows Subversion Client

Install TortoiseSVN, the free Subversion client from Tigris. The website is <http://tortoisesvn.net/>

To download Alien Arena from the SVN Repository:

- In Windows Explorer, go to the *Computer # C: folder*.
- Right click in the window and select *SVN Checkout* in the context menu.
- For the *URL of repository*, enter: *svn://icculus.org/alienarena/trunk*.
- For the *Checkout directory*, enter: *C:\alienarena\_w32*.
- Leave the *Checkout Depth* at *Fully Recursive*, *Omit Externals* unchecked, and *Revision* at *Head Revision*.
- Click *OK*, and let TortoiseSVN do its thing.

## 3.2. Microsoft Visual C++ 2010 Express

Microsoft Visual C++ 2010 Express is part of Microsoft Visual Studio 2010 and is provided free of charge. Registration is required. Download Visual C++ 2010 Express from the Microsoft website (<http://www.microsoft.com/>) and install it using the default setups.

Running programs compiled with VC++ 2010 requires the redistributable dll, *msvcr100.dll*. Some systems may not have it installed; a copy is provided in SVN.

## 3.3. Libraries

Extract the libraries in *lib\_zipfiles* to *VS2010\libsrc*.

### Patch for curl-7.20.1.

Unzip *libcurl\_patch.zip* and replace *curl-7.20.1/lib/setup\_once.h*. The original file, renamed *setup\_once\_unpatched.h*, is included in the *.zip* for reference.

### Using premake for ODE config

A version of premake that supports VC++ 2010 is available at <http://industriousone.com/premake>.

(TODO: add instructions for running *premake4* with the proper options).

### Some commentary on libraries

There is a rationale for keeping the libraries in the original *.zip* archives in SVN, rather than extracting them into SVN. For one thing, they are large and it would add a lot of clutter to the repository. Also, the build is setup to allow for multiple co-existing versions when libraries are updated. It would be good to be able to bring in a new version's *.zip* into SVN and get it working before making the switch in the official build.

All the libraries supply build procedures for a variety of environments. But, most did not have VC10 builds at the time they were setup. The solution was to create projects in Visual Studio for each of them. This has advantages, but is more complicated than just using a supplied build. It does take some work to get the build details correct and consistent.

## 3.4. OpenAL 1.1 SDK

Download the OpenAL SDK from OpenAL.org: <http://www.openal.org/>. (Redirects to the CreativeLabs site.) In the Download section, get the *OpenAL 1.1 Core SDK*. Install it in the default location; the build procedure is setup to look for it there.

### Caution

On an x86 system, for example an old XP box, you will need to change the path to use "Program files (x86)" instead of the default.

Run the *oalinst.exe* program to setup the OpenAL driver, if you have not done that previously.

The *oalinst.exe* program installs whatever is required for OpenAL to use a particular systems sound hardware. It is included in the Alien Arena Windows distribution and is in SVN.

## 3.5. Directory Tree

Here are the directories and files related to the Windows build.

alienarena\_w32\

alienarena\_w32.sln

Solution file. Mostly just a list of the Projects in VS2010\.

alienarena\_w32.props

Project property file. Generated and edited with the Visual Studio *Property Manager*. It is fairly easy, and sometimes more convenient, to manually edit it. Projects can inherit compiler options, etc. from here; which makes it much easier to make changes that affect all projects.

msvcr100.dll

Redistributable for Visual C++ 2010.

oalinst.exe

OpenAL installer.

vs2010\

Build directory for Windows version of Alien Arena.

Project Files

Project files. Generated and edited with Visual Studio. Somewhat easy to manually edit.

- crx.vcxproj
- game.vcxproj
- freetype.vcxproj
- libcurl.vcxproj
- libjpeg.vcxproj
- libode.vcxproj
- libogg.vcxproj
- libopcode.vcxproj
- libvorbis.vcxproj
- libvorbisfile.vcxproj
- zlib.vcxproj

bin\

Target directory. The *alienarena.exe* executable is placed here. There are *RELEASE* and *DEBUG* subdirectories created by the build procedure.

lib\

Target directory. The game and other libraries are placed here. There are *RELEASE* and *DEBUG* subdirectories created by the build procedure.

libsrc\

Source directory for libraries.

include\

Header files. The build procedure copies header files from the *libsrc* subdirectories to these directories for most of the libraries.

- config.h — *config.h* for *alienarena.exe* and *game.lib*.
- curl\
- jpeg\
- ode\
- ogg\
- vorbis\

- freetype-config\
  - ftconfig.h
  - fthead.h
  - ftmodule.h
  - ftoption.h
  - ftstdlib.h

The build procedure populates the *include* directories for the various libraries. Makes the build more complicated, but should make it easier to work with a new library version in parallel with a current one.

## 3.6. The config.h file

The *config.h* file for *alienarena.exe* and *game.lib* is an artifact from the Unix/Linux build. In Unix/Linux, *config.h* is generated by the *configure* procedure. For Windows, it is manually edited.

## 4. Unix/Linux Build

### 4.1. Introduction

Beginning in August of 2010 the Unix/Linux build for Alien Arena was converted from a Makefile to Autotools. There are two main goals for the implementation: One, to improve support for the various Linux distributions' package maintenance process. The other, to improve portability and ease of installation for users.

It is important to recognize that the goal of GNU Autotools is NOT to make the developer's job easier. From a coder's viewpoint, it is not, nor is it intended to be, the ideal software construction tool. The mere mention of Autotools elicits groans from most developers. There are alternatives, but for open source software, none have developed to the point of being generally adopted as standard. There are some who disagree with that statement, but they are mistaken.

For development purposes it is sometimes better to use a plain makefile. See *Developer's Not-Autotools Makefile* below.

### Reading List for Autotools

- John Calcote. *Autotools: A Practitioner's Guild to GNU Autoconf, Automake, and LibTool*.
- Diego E. "Flameeyes" Petteno. *Autotools Mythbuster* <http://www.flameeyes.eu/autotools-mythbuster/>.
- David MacKenzie, Ben Elliston, Akim Demaille. *Autoconf: Creating Automatic Configuration Scripts* For version 2.65, 4 November 2009.
- David MacKenzie, Tom Tromey, Alexandre Duret-Lutz. *GNU Automake*. For version 1.11.1, 8 December 2009.
- Richard M. Stallman, Roland McGrath, Paul D. Smith. *GNU Make: A Program for Directing Recompilation*. GNU make Version 3.81, April 2006.
- Gordon Matzigkeit, Alexandre Oliva, Thomas Tanner, Gary V. Vaughan. *GNU Libtool*. For version 2.2.6, 1 August 2008.
- Richard Stallman, et al. *GNU Coding Standards*. last updated June 10, 2008.
- Edited by Rusty Russell, Daniel Quinlan, Christopher Yeoh. *Filesystem Hierarchy Standard* Filesystem Hierarchy Standard Group.
- Gary V. Vaughan, Ben Elliston, Tom Tromey and Ian Lance Taylor. *GNU AutoConf, AutoMake, and LibTool "The Goat Book"*, Version 1.5, February 2006. <http://sourceware.org/autobook>.

### 4.2. Program Changes in 7.50

Previous versions had support for configurable *DATADIR* and *LIBDIR* installation paths. However, the implementation was not complete; due largely to the file system organization inherited from the Quake source. The installation procedure now uses configure arguments for determining installation paths.

Linux distributions install the game in the */usr* hierarchy in conformance with Unix file system standards. The */usr* hierarchy requires root privileges for writing, of course. While the game already had support for a user writeable directory, *.codeded/*, in the users home directory, it did not implement a place to write bot information. Therefore, support for a *botinfo* directory in the user's home directory was added.

To simplify the installation and to avoid the complications of using a shared library, the game module shared library, *game.so*, is now a static library. As a result, the *LIBDIR* installation variable is eliminated. Internally, the interface to the game module retains the dynamic link structure. Thus, it is still possible, in theory, to have a dynamic *game.so* that overrides the statically linked version. This is not tested nor supported in version 7.50.

Two new functions were added to the server/game interface for file system access. The location of game data files depends on how the game is installed. The code should not make assumptions about the base path since it might be affected by "prefix" and "datadir" configuration options. The game module will now determine all file paths using those functions (eliminating problems where the code assumed the legacy Quake style file system). The additional functions make it possible to place a *botinfo* directory, in user-writeable file space. The two new functions supply the full filesystem path when given a relative path. The functions are **FullPath()** and **FullWritePath()**. They are declared in *game/game.h* and *qcommon/qcommon.h*. They are defined in *qcommon/files.c*.

The name of the stand-alone dedicated server was changed from *crded* to *crx-ded*. The purpose was to make it simple to change the name of the executable files using the standard built-in configure option. With the 7.60 release, the dedicated server name was updated to *alienarena-ded*.

## 4.3. Subversion Repository

### Linux Subversion Client

Install the free Apache (formerly Tigris) Subversion client using your package manager. Or download it from the website at <http://subversion.apache.org/>.

To download Alien Arena from the SVN Repository:

- Create a '*PATH*' for installation somewhere in your home directory.
- Run: `svn checkout svn://icculus.org/alienarena/trunk PATH`.

For example:

```
cd ~ & mkdir alienarena & svn checkout svn://icculus.org/alienarena/trunk alienarena
```

## 4.4. Maintainer's Guide

### Autotools Source Files

These are Autotools-related files that are manually edited:

`configure.ac`  
source file for *autoconf*.



**Makefile.am**

top level source file for *automake*.

**game\_data.am**

lists of game asset files. included in *Makefile.am*.

**source/Makefile.am**

*source* subdirectory source file for *automake*

Some Autotools files are just copied without change:

**m4/\*.m4**

m4 macros from the autoconf-archive.

**INSTALL**

GNU build/install reference.

**COPYING**

GNU GPL, Version 2.

## Autotools Generated Files

The Alien Arena Autotools build uses **Maintainer Mode**. The upside of this is that the end user does **not** need the Autotools programs and can run the well-known *./configure, make, sudo make install* procedure. The downside is that if *configure* is run without *--enable-maintainer-mode*, some of the generated files will not be updated.

The **Generated Files** are:

**configure**

the user's configure script.

**Makefile.in**

template for the user's top directory Makefile.

**source/Makefile.in**

template for the user's source subdirectory Makefile.

**config/config.h.in**

template for the user's config.h.

**aclocal.m4**

all the .m4 macros used by *configure*. generated by *aclocal*.

The *Generated Files* need to be regenerated when any of following occur:

- Source files (.c, .h, etc.) are added, removed or renamed.
- System libraries are added or removed (in *configure.ac*, *Makefile.am* files).
- Asset files (textures, models, etc.) are added, removed or renamed.
- Autotools files (*configure.ac*, *Makefile.am*, etc.) are modified.

**Note** There are other intermediate files generated by Autotools. Refer to Autotools documentation.

## Autotools Prerequisites

The Autotools packages are:

- *autoconf*
- *automake*,
- *autoconf-archive*

To avoid confusion, developers committing to the repository should all be using the same versions of these packages.

The *autoconf-archive* is an evolving set of macros maintained on the GNU website. We use a few of these. The file names are normally prefixed with *ax\_* and the macros prefixed with *AX\_*. Each file has a *serial* number in a comment. Care should be taken not to commit an earlier version of a *.m4* file into the SVN *m4* subdirectory. If a file we use is updated in the archive, update it in the repository.

After adding new *AX\_* macros to *configure.ac*, the *.m4* file needs to be added to the *m4* subdirectory. This is done with the following command.

```
aclocal -I m4 --force --install
```

## Adding, Removing, Renaming Program Source Files

Source files are modified in the *source/Makefile.am* file. Modify the source in the applicable *\*\_SOURCE* lists. Header (*.h*) files must be included, because the list is used to generate the tarball.

Certain files may appear in several lists. For example, *'alienarena\_SOURCES'*, *alienarena-ded\_SOURCES* and *libgame\_a\_SOURCES* all have *qcommon/qfiles.h*.

## Adding, Removing, Renaming Game Asset Files

The file, *game\_data.am* contains lists of game asset files. It is included in the top-level *Makefile.am*. It is used to generate the distribution archive and for copying files to the *datadir* in a *standard install*.

The downside is that *game\_data.am* needs to be kept up-to-date with additions and deletions. The advantage is that the *install* program handles the copying of game resources. For *make uninstall* it handles deletion of what *make install* installed.

To make modifications, find the appropriate list and add, delete or modify the file entry. Makefile syntax requires that there be no trailing spaces after the *\*. Each entry should be preceded by one TAB character. The files are listed in (mostly) alphabetical order.

### Version 7.60 changes to *game\_data.am*

With the game resource additions of 7.60, the *game\_data.am* list exceeded the command line length limit. After much agonizing and lamenting, a solution was found. The list is subdivided by the subdirectories of *data1/*. This works ok for the *install* procedure. Unfortunately, it does not work for the distribution creation procedure. For that, a copy procedure is added to *dist-hook* in *Makefile.am*.

## Is this a PITA, or what?

Yes, it is. It would be nice not to have to reconfigure when game assets are added or deleted. The way to avoid this is to implement a custom *install* procedure that uses the game data file list to copy files when installing and deleting files when uninstalling. The game data file list would also be used for distribution creation. The method in version 7.60 is crude; there is probably a faster way. (However, speed is not important for tarball generation.)

It would be good to support separate source and data tarballs. This would make us more popular with distro packagers. Some distros split the tarball, possibly because the source code and game assets are under different licenses.

## Adding or Removing System Libraries

The details of changing the system library settings are way beyond the scope of this document. Some knowledge of *autoconf* and *automake* is required.

For common system libraries, there **may** be a helpful *.m4* macro available. These may be part of *autoconf* (*AC\_\**), or they may be extensions from the Autoconf Archive (*AX\_\**).

## Procedures

There are various clean options, besides the usual, that remove more than object files. Some useful ones are:

*maintainer-clean*

removes most generated files. run *autoreconf* to rebuild them.

*distclean*

removes files generated by *configure*.

Following changes, run:

- `autoreconf --verbose`
- `./configure --enable-maintainer-mode <other options>`
- `make`
- `sudo make install`

There are variations for the *configure* and *make install* commands, of course. See below for *configure* options and compiler options.

Run some tests and, if all this goes well, commit the changed files to the repository. The *svn status* command will show which files were modified.

## Installation Notes

For the *standard install*, use *make install-exec* to limit copying to the executables. Note that the *alternate install* always just copies the executables.

## 4.5. Packager's Guide

### Verify Autotools Versions

Verify that your versions of *autoconf*, *automake*, and *autoconf-archive* are current.

The 7.60 release uses *autoconf* 2.68 and *automake* 1.11. If more than one dev is committing autotool related changes to SVN, it will be less confusing if all are using the same versions.

### Verify Alien Arena Version

Verify that the version number in *configure.ac* is correct.

### Verify Autotools Files

See the *Maintainer's Guide* above. Verify that all Autotools-related files are up-to-date.

### Export from Subversion Repository

Export from the SVN trunk to a directory for the distribution build.

Verify that the files in *config/* and *m4/* subdirectories are current. One way to do this is to rename the *config/* and *m4/* directories and then run:

```
aclocal --force --install -I m4
```

```
autoreconf --force --install
```

Compare the files in the new *config/* and *m4/* with the previous ones. If they match, proceed. If not, then the files in SVN need to be updated. The cleanest thing to do would be to update SVN and do a new export.

### Generate the Distribution Archive

Run this command to build and verify the distribution package:

```
make distcheck
```

If this succeeds, the *alienarena-<version>.tar.gz* has been created.

### Test the Distribution Package

In a test directory, extract the distribution archive. Run the configure, make, make install sequence. Use the *--prefix* configure option to install into a test directory.

To test the dedicated server only build, create a subdirectory and build with an alternative test directory. In the subdirectory, *configure* is invoked with *../configure*.

For completeness, the alternate install should also be tested.

## Example

An example of distribution package generation:

- With a development directory in *\$HOME/alienarena*.
- Using standard install into */usr/local*.

```
$ cd ~
$ mkdir aadist
$ cd $HOME/alienarena
$ svn export . $HOME/aadist/aaexport
$ cd $HOME/aadist/aaexport
$ ./configure
$ make distcheck
$ cd ..
$ mkdir aainstall
$ cd aainstall
$ cp ../aaexport/alienarena-7.50.tar.gz .
$ tar -xzf alienarena-7.50.tar.gz
$ cd alienarena-7.50
$ ./configure
$ make
$ sudo make install
$ cd ~
$ alienarena
```

## BASH script for distribution tarball generation

This script uses several SVN *export* commands to retrieve the subset of files needed for the release tarball. It then invokes *make dist-check* to build the tarball. Run the script in a RELEASE directory.

The following commands generate the checksums and then run a *diff* to check for possibly missing files.

In the RELEASE directory:

```
$ mv alienarena/alienarena-<version>.tar.gz .
$ md5sum alienarena-<version>.tar.gz >md5
$ shasum alienarena-<version>.tar.gz >shal
$ tar -xzf alienarena-<version>.tar.gz
$ diff -r -q alienarena alienarena-<version>
```

The Script:

```
#!/bin/bash
#
# Alien Arena release tarball generation
#
# --- top level ---
svn export --ignore-externals --non-recursive \
  svn://icculus.org/alienarena/trunk alienarena
###
# --- config/ ---
svn export --ignore-externals \
  svn://icculus.org/alienarena/trunk/config alienarena/config
###
```

```
# --- m4/ ---
svn export --ignore-externals \
  svn://icculus.org/alienarena/trunk/m4 alienarena/m4
###
# --- source/ ---
svn export --ignore-externals \
  svn://icculus.org/alienarena/trunk/source alienarena/source
###
# --- docs/ ---
svn export --ignore-externals \
  svn://icculus.org/alienarena/trunk/docs alienarena/docs
###
# --- Tools/ (fuse.tar.gz only) ---
svn export --ignore-externals --non-recursive \
  svn://icculus.org/alienarena/trunk/Tools alienarena/Tools
###
# --- Tools/LinuxScripts/ ---
svn export --ignore-externals \
  svn://icculus.org/alienarena/trunk/Tools/LinuxScripts alienarena/Tools/LinuxScripts
###
# --- arena/ ---
svn export --ignore-externals \
  svn://icculus.org/alienarena/trunk/arena alienarena/arena
###
# --- botinfo/ ---
svn export --ignore-externals \
  svn://icculus.org/alienarena/trunk/botinfo alienarena/botinfo
###
# --- data1/ ---
svn export --ignore-externals \
  svn://icculus.org/alienarena/trunk/data1 alienarena/data1
###
# --- tarball construction ---
cd alienarena
./configure --enable-maintainer-mode
make distcheckDeveloper's Not-Autotools Makefile
```

## 4.6. Configure options

Run `./configure --help` to see all *configure* options. These are *configure* options added for Alien Arena.

Post 7.60 release, certain debug options have been removed. Some were obsolete, but mostly this is a "philosophical" issue: debug options, including compiler warning options, may be confusing to the non-developer. For developers, specifying CFLAGS is more flexible. (Note: when the debug options were added, strat questioned these but deferred to blackice's greater expertise with autotools. But time has shown them to be an unnecessary complication.)

`--disable-client`

build dedicated server only, not the client (default:no)

`--enable-alternate-install`

traditional single directory, in-place installation (default:no)

`--enable-ansi-color`

ANSI terminal color (default: no) Added in 7.51.

`--disable-build-status`

hide the status message at the end of the configuration script (default:no)

`--with(out)-xf86vm`

include XF86 VidMode support (default: check) Note: Needed to support full-screen mode. Note: If *with* is specified, then the library is required.

`--with(out)-xf86dga`

include XF86 DGA support (default: without) Note: default is changed from *check* in version 7.60 Note: DGA appears to only used for mouse pointer input and is optional. It is disabled by default, or when the cvar, *in\_dgmouse*, is set to 0. Note: If *with* is specified, then the library is required.

`--with(out)-zlib`

include Zlib compressDeveloper's Not-Autotools Makefileion support (default: check) Note: If *with* is specified, then the library is required.

`--with-icondir=DIR`

icon install directory (default:DATADIR/icons)

## 4.7. GCC Compiler Options

Linux distro's use a variety of different gcc compile options. These can give more error warnings; and in rare cases program errors.

Also for users who compile from source, there are CPU dependent options that might give better performance. Might be good to document some of these in the README.

Wall , Wextra

With -Wextra many warnings are produced, especially for unused parameters and signed/unsigned comparisons. With just -Wall, a few unused variable warnings are produced, which are easily fixed. Some bogus uninitialized warnings are produced; it is possible, but annoying, to eliminate these.

O2 , O3

Optimization levels. O3 produces different warning messages, so should be tested before releases.

m32 , m64

Specify 32bit or 64bit build. Only needed when cross-compiling.

march , mtune

Set for specific architectures. Might give better performance than generic builds. For instance, -march=core2 enables higher performance features. Use -mtune=generic when building release binaries; for Desura, for instance.

mfpmath

Testing shows that -mfpmath=sse gives better performance when compiling with GNU gcc.

FORTIFY\_SOURCE

-Wp,D\_FORTIFY\_SOURCE=2 part of "stack smashing" protection

fexceptions

TBD.

fstack-protector

Part of "stack smashing" protection.

param

--param=ssp-buffer-size=4. Part of "stack smashing" detection.

fasynchronous-unwind-tables

From fedora. TBD.

## 4.8. config.h

The config.h file is auto-generated for configurable builds (e.g Linux). It is manually edited for non-configurable builds (e.g. Windows). The *configure.ac* file does generate Windows related definitions in *config.h* so it can be used as a basis for the Windows *config.h*.

Rather than use symbols built into the compiler these symbols (and some others related to targets) are defined: *WIN32\_VARIANT* and *UNIX\_VARIANT*. This should make it easier to keep system dependent variations organized. It is a good idea, of course, to minimize system dependent conditional compilation in the common code.

## 4.9. Open Dynamics Engine (ODE) Integration

A static library, *libode.a*, is built and linked with the main program, *alienarena*. Only files that are used in the Developer's Not-Autotools Makefilelibrary build are included in the source tree in SVN and in the distribution. There is only one *config.h* and it includes what is needed for building *libode.a*.

The ODE update to version 0.12 removed a function that we use. As of this writing there is no alternative. So ODE 0.12 cannot be used, yet. So the build with ODE 0.11 integrated is preferred.

## 5. MinGW Build

Experimentally, Alien Arena has been built using MinGW using the Autotools build. In the future, details and procedures may appear here.

## 6. Mac OS X / Darwin X11 Build

The Autotools build has experimental support for Mac OS X using the Darwin X11 environment. There are reports of some progress with this using MacPorts and Homebrew.

In February 2011, a native port to Mac OS X was developed. Details and test results to be reported as they become available.

## 7. Developer's Not-Autotools Makefile

In the *docs/* directory, the file, *dev-makefile*, is a *Makefile* intended to support experimental builds. Its primary purpose is to allow custom builds with additional or different sources without having to hack the Autotools build. Documentation on how to customize it is included within the *dev-makefile* file.



It is simpler to make a full copy of the trunk sources, but that is not necessary. It takes some trickiness setting up *vpaths* and customizing *-I* options, but it is possible to do an alternate build with a small subset of sources in an alternate source subdirectory. Good for experimenting with optimizations.

## 8. Desura

TODO:

## 9. Static Analysis

TODO:

## 10. Phoronix Test Suite

TODO: