Arch User Repository

Summary

The Arch User Repository is a collection of user-submitted PKGBUILDs that supplement software available from the official repositories. This article describes how to buildunsupported software packages from the AUR.

Overview

Packages in Arch Linux are built using makepkg and a custom build script for each package (known as a PKGBUILD). Once packaged, software can be installed and managed with pacman. PKGBUILDs for software in the official repositories are available from the ABS tree; thousands more are available from the (unsupported) Arch User Repository.

The Arch User Repository (AUR) is a community-driven repository for Arch users. It contains package descriptions (PKGBUILDs) that allow you to compile a package from source with makepkg and then install it via pacman. The AUR was created to organize and share new packages from the community and to help expedite popular packages' inclusion into the community repository. This document explains how users can access and utilize the AUR.

A good number of new packages that enter the official repositories start in the AUR. In the AUR, users are able to contribute their own package builds (PKGBUILD and related files). The AUR community has the ability to vote for or against packages in the AUR. If a package becomes popular enough -- provided it has a compatible license and good packaging technique -- it may be entered into the community repository (directly accessible by pacman or abs).

Getting started

Users can search and download PKGBUILDs from the AUR Web Interface. These PKGBUILDs can be built into installable packages using makepkg, then installed using pacman.

Read the remainder of this article for more info and a short tutorial on installing AUR packages.

Visit the AUR Web Interface to inform yourself on updates and happenings. There you will also find statistics and an up-to-date list of newest available packages available in AUR.

Glance over the #FAQ for answers to the most common questions.

You may wish to adjust /etc/makepkg.conf to better optimize for your processor prior to building packages from the AUR. A significant improvement in compile times can be realized on systems with multi-core processors by adjusting the MAKEFLAGS variable. Users can also enable hardware-specific optimizations in GCC via the CFLAGS variable. See makepkg.conf for more information.

Install "base-devel" (pacman -S base-devel), because members of this group are not explicitly required by AUR packages which may not build without them (more info in this thread).

History

The following items are listed for historical purposes only. They have since been superseded by the AUR and are no longer available.

At the beginning, there was ftp://ftp.archlinux.org/incoming, and people contributed by simply uploading the PKGBUILD, the needed supplementary files, and the built package itself to the server. The package and associated files remained there until a Package Maintainer saw the program and adopted it.

Then the Trusted User Repositories were born. Certain individuals in the community were allowed to host their own repositories for anyone to use. The AUR expanded on this basis, with the aim of making it both more flexible and more usable. In fact, the AUR maintainers are still referred to as TUs (Trusted Users).

Searching

The AUR web interface can be found here, and an interface suitable for accessing the AUR from a script (for example) can be found here Queries search package names and descriptions via a MySQL LIKE comparison. This allows for more flexible search criteria (e.g. try searching for 'tool%like%grep' instead of 'tool like grep'). If you need to search for a description that contains '%', escape it with '\%'.

Installing packages

Installing packages from the AUR (aka the unsupported repository) is a relatively simple process. Essentially:

1. Acquire the tarball which contains the PKGBUILD and possibly other required files
2. Extract the tarball (preferably in a folder set aside just for builds from the AUR)
3. Run makepkg in the directory where the files are saved ("makepkg -s" will auto-resolve dependencies with pacman)
4. Install the resulting package with pacman

```
# pacman -U /path/to/pkg.tar.xz
```

AUR Helpers add seamless access to the AUR. They vary in their features, but can ease in searching, fetching, building, and installing from PKGBUILDs found in the AUR. All of these scripts can be found in UNSUPPORTED.

Note: There is not and will never be an official mechanism for installing build material from UNSUPPORTED. All users should be familiar with the build process.

What follows is a detailed example of installation of a package called "foo".

Prerequisites

First ensure that the necessary tools are installed. The package group "base-devel" should be sufficient; it includes make and other tools needed for compiling from source.

Warning: Packages in the AUR assume "base-devel" is installed, and will not list members of this group as dependencies even if the package cannot be built without them. Please ensure this group is installed before complaining about failed builds.

```
# pacman -S base-devel
```

Next choose an appropriate build directory. A build directory is simply a directory where the package will be made or "built" and can be any directory. Examples of commonly used directories are:

```
~/builds
```

or if using ABS (the Arch Build System):

```
/var/abs/local
```

For more information on ABS read the Arch Build System article. The example will use ~/builds as the build directory.

Acquire build files

Locate the package in the AUR. This is done using the search feature (text field at the top of the AUR home page). Clicking the application's name in the search list brings up an information page on the package. Read through the description to confirm that this is the desired package, note when the package was last updated, and read any comments.

Download the necessary build files. From the package's information page download the build files by clicking the "Tarball" link on the left-hand side near the end of the package details. This file should be saved to the build directory or otherwise copied to the directory after downloading. In this example, the file is called "foo.tar.gz" (standard format is <pkgname>.tar.gz, if it has been properly submitted).

Build the package

Extract the tarball. Change directories to the build directory if not already there and extract the build files.

```
$ cd ~/builds
$ tar -xvzf foo.tar.gz
```

This should create a new directory called "foo" in the build directory.

Warning: Carefully check all files. Change directories to the newly created directory and carefully check the PKGBUILD and any .install file for malicious commands. If in doubt, do NOT build the package and seek advice on the forums or mailing list.

```
$ cd foo
$ nano PKGBUILD
$ nano foo.install
```

Make the package. After manually confirming the integrity of the files, run makepkg as a normal user in the build directory.

```
$ makepkg -s
```

The -s switch will use sudo to install any needed dependencies. If the use of sudo is undesirable, manually install required dependencies beforehand and exclude the -s in the above command.

Install the package

Install the package using pacman. A tarball should have been created named:

<application name>-<application version number>-<package revision number>-<architecture>.pkg.tar.xz

This package can be installed using pacman's "upgrade" command:

```
# pacman -U foo-0.1-1-i686.pkg.tar.xz
```

Note: The above example is only a brief summary of the package building process. A visit to the makepkg and ABS pages will provide more detail and is highly recommended (particularly for first-time users).

Sharing packages

The user plays an essential role in the AUR, which cannot fulfill its potential without the support, involvement, and contribution of the wider user community. The life-cycle of an AUR package starts and ends with the user and requires the user to contribute in several ways.

Users can share PKGBUILDs using the UNSUPPORTED area in the AUR. UNSUPPORTED does not contain any binary packages but allows users to upload PKGBUILDs that can be downloaded by others. These PKGBUILDs are completely unofficial and have not been

thoroughly vetted, so they should be used at your own risk.
Feedback
A comments facility allows users to provide suggestions and feedback on improvements to the PKGBUILD contributor. Avoid pasting patches or PKGBUILDs into the comments section. They quickly become obsolete and just end up needlessly taking up lots of space. Instead email those files to the maintainer, or even use a pastebin like http://aur.pastebin.com.
One of the easiest activities for all Arch users is to browse the AUR and vote for their favorite packages using the online interface. All packages are eligible for adoption by a TU for inclusion in community, and the vote count is one of the considerations in that process; it is in everyone's interest to vote!
Submitting packages
After logging in to the AUR web interface, a user can submit a gzipped tarball (.tar.gz) of a directory containing build files for a package. The directory inside the tarball should contain a PKGBUILD, any .install files, patches, etc. (ABSOLUTELY no binaries). Examples of what such a directory should look like can be seen inside /var/abs if ABS was installed.
The tarball can be created with the following command:
$ makepkg --source
Note that this is a gzipped tarball; assuming you are uploading a package called libfoo, when you create the file it should look similar to this:
# List contents of tarball.
$ tar tf libfoo-0.1-1.src.tar.gz
libfoo/
libfoo/PKGBUILD
libfoo/libfoo.install
When submitting a package, observe the following rules:
Check core, extra, and community for the package. If it is inside any of those repositories in ANY form, DO NOT submit the package (if the current package is broken or is lacking an included feature then please file a bug report in FlySpray).
Check AUR for the package. If it is currently maintained, changes can be submitted in a comment for the maintainer's attention. If it is unmaintained, the package can be adopted and updated as required.
Verify carefully that what you are uploading is correct. All contributors must read and adhere to the Arch Packaging Standards when writing PKGBUILDs. This is essential to the smooth running and general success of the AUR. Remember you are not going to earn any credit or respect from your peers by wasting their time with a bad PKGBUILD.
Packages that contain binaries or that are very poorly written may be deleted without warning.
If you are unsure about the package (or the build/submission process) in any way, submit the PKGBUILD to the AUR Mailing List or the AUR boards on the forum for public review before adding it to the AUR.
Make sure the package is useful. Will anyone else want to use this package? Is it extremely specialized? If more than a few people would find this package useful, it is appropriate for submission.
Gain some experience before submitting packages. Build a few packages to learn the process and then submit.
If you submit a package.tar.gz with a file named 'package' in it you'll get an error: 'Could not change to directory /home/aur/unsupported/package/package'. To resolve this, rename the file named 'package' to something else, for example, 'package.rc'. When it is installed in the pkg directory you may rename it back to 'package'.

Maintaining packages
If you maintain a package and want to update the PKGBUILD for your package just resubmit it.
Check for feedback and comments from other users and try to incorporate any improvements they suggest; consider it a learning process!
Please DO NOT just submit and forget about packages! While in UNSUPPORTED, it is the user's job to maintain the package by checking for updates and improving the PKGBUILD.
If you do not want to continue to maintain the package for some reason, disown the package using the AUR web interface and/or post a message to the AUR Mailing List.
community
The community repository, maintained by Trusted Users, contains the most popular packages from AUR. It is enabled by default in pacman.conf. If disabled/removed, it can be enabled by uncommenting/adding these two lines:
File: /etc/pacman.conf
...
community
Include = /etc/pacman.d/mirrorlist
...
community, unlike AUR, contains binary packages that can be installed directly with pacman and the build files can also be accessed with ABS. Some of these packages may eventually make the transition to the core or extra repositories as the developers consider them crucial to the distribution.
Users can also access the community build files by editing /etc/abs.conf and enabling the community repository in the REPOS array.
Git Repo
A Git Repo of the AUR is maintained by Thomas Dziedzic providing package history among other things. It is updated at least once a day.
To clone the repository (several hundred MB):
$ git clone git://pkgbuild.com/aur.git
Web interface, Readme, Forum thread

FAQ
Q: What is the AUR?
A:The AUR (Arch User Repository) is a place where the Arch Linux community can upload PKGBUILDs of applications, libraries, etc., and share them with the entire community. Fellow users can then vote for their favorites to be moved into the community repository to be shared with Arch Linux users in binary form.
Q: What is a TU?
A:A TU (Trusted User) is a person who is chosen to oversee AUR and the community repository. They're the ones who maintain popular PKGBUILDs in community, and overall keep the AUR running.
Q: What's the difference between unsupported and community?
A:unsupported is where all PKGBUILDs that users submit are stored, and must be built manually with makepkg. When PKGBUILDs receive enough votes, they are moved into the community repository, where the TUs maintain binary packages that can be installed with pacman.
Q: How many votes does it take to get a PKGBUILD into community?
A:Usually, at least 10 votes are required for something to move into community. However, if a TU wants to support a package, it will often be found in the repository.
Q: How do I make a PKGBUILD?
A:The best resource is Creating Packages. Remember to look in AUR before creating the PKGBUILD as to not duplicate efforts.
Q: I'm trying to do pacman -S foo; it isn't working but I know it's in community
A:You probably haven't enabled community in your /etc/pacman.conf. Just uncomment the relevant lines. If community is enabled in your /etc/pacman.conf try running pacman -S -y first to synchronize the pkgcache before trying your package again.
Q: Foo in AUR is outdated; what do I do?
A:For starters, you can flag packages out-of-date. If it stays out-of-date for an extended amount of time, the best thing to do is email the maintainer. If there is no response from the maintainer, you could mail to the aur-general mailing list to have a TU orphan the PKGBUILD if you're willing to maintain it yourself.
Q: I have a PKGBUILD I would like to submit; can someone check it to see if there are any errors?
A:If you would like to have your PKGBUILD critiqued, post it on the aur-general mailing list to get feedback from the TUs and fellow AUR members. You could also get help from the IRC channel, #archlinux on irc.freenode.net. You can also use namcap to check your PKGBUILD and the resulting package for errors.
Q: Foo in AUR doesn't compile when I do makepkg; what should I do?
A:You are probably missing something trivial.
1. Run pacman -Syyu before compiling anything with makepkg as the problem may be that your system is not up-to-date.
2. Ensure you have both "base" and "base-devel" groups installed.
3. Try using the "-s" option with makepkg to check and install all the dependencies needed before starting the build process.
The reason might not be trivial after all. Custom CFLAGS, LDFLAGS and MAKEFLAGS can cause failures. It's also possible that the PKGBUILD is broken for everyone. If you can't figure it out on your own, just report it to the maintainer.
Q: How can I speed up repeated build processes?
A:If you frequently compile code that uses gcc - say, a git or SVN package - you may find ccache, short for "compiler cache", useful.
Q: How do I access unsupported packages?
A:See #Installing packages
Q: How can I upload to AUR without using the web interface?
A:You can use aurploader or burp -- both with command-line interfaces.