[Date Prev][Date Next][Thread Prev][Thread Next][Date Index][Thread Index]

Symbol stripping tips + handy script

Subject: **Symbol stripping tips + handy script**
From: "Andy O'Meara" <email@hidden>
Date: Tue, 21 Mar 2006 17:24:46 -0500
Delivered-to: email@hidden
Delivered-to: email@hidden
Thread-index: AcZNNkImgNRoGbkpEdqCWwAKlaBZUA==
Thread-topic: Symbol stripping tips + handy script
User-agent: Microsoft-Entourage/11.2.1.051004

Title: ***Symbol stripping tips + handy script***

Rob Barris and I have put together a handy post build phase script that many people here may find useful. The script ideal for folks here that have concerns over binary size and code secrecy/security but don't have enough hours in the day to learn the ins and outs of Xcode/gcc/ld/strip. The comments of the script say the rest...

Simply add the following text as a post build phase script in Xcode...

```
# /usr/bin/perl -w
#
# Last updated: 21MAR06, Andy O'Meara and Rob Barris
#
# This is an Xcode post-build phase script for devs who sleep better at night knowing
#    that their deployment binaries are as stripped as possible.  This makes life more difficult
#    for a hacker/cracker to locate sensitive code to trace, study, and/or extract.
#
# This script will execute only if the Xcode "Deployment Postprocessing" setting
#    is set (aka DEPLOYMENT_POSTPROCESSING).
#
# The downside to shipping a stripped binary is that your user's crash reports
#    will be useless unless you have a link map to convert code offsets (from a stack trace)
#    into proc names.  To address this, this script moves your pre-stripped executable
#    to the build dir, appending "_full" to the filename, allowing you to retain it for
#    the day you need it in order to decipher a stack trace.  You do this by using 'atos'
#    with the original generated binary (type 'man atos' for info).
#
# Recommended Xcode build settings:
#    Dead Code Stripping                YES
#    Only Link In Essential Symbols     NO
#    Deployment Postprocessing          YES (this activates this script)
#    Strip Linked Product               NO
#    Use Separate Strip                 NO
#    Strip Style                        All Symbols
#    Strip Debug Symbols During Copy    NO
#    Preserve Private External Symbols  NO
#    Separate PCH Symbols               YES
#    Symbols Hidden By Default          YES (Critical!)
#    Inline Functions Hidden            YES
#
# Note that if you're building a dynamic library, you'll need to explicitly
#    declare any symbols that you want to be exported.  See the following:
#    file:///Developer/ADC Reference Library/documentation/DeveloperTools/Conceptual/CppRuntimeEnv/Articles/SymbolVisibility.html
#

use strict;

die "$0: Must be run from Xcode" unless $ENV{"BUILT_PRODUCTS_DIR"};

# This script is activated via an Xcode env flag.
if ( $ENV{DEPLOYMENT_POSTPROCESSING} ne "YES" ) {
   exit 0;
}

print "\n\n==================== Commencing external stripping phase...\n";

my $BINARY       = "$ENV{BUILT_PRODUCTS_DIR}/$ENV{WRAPPER_NAME}/Contents/MacOS/$ENV{EXECUTABLE_NAME}";
my $BINARY_FULL  = "$ENV{BUILT_PRODUCTS_DIR}/$ENV{EXECUTABLE_NAME}_full";
my $BINARY_i386  = "${BINARY}_i386";
my $BINARY_ppc   = "${BINARY}_ppc";


# Extract each arch into a "thin" binary for stripping
`lipo "$BINARY" -thin  ppc -output "$BINARY_ppc" `;
```

```
`lipo "$BINARY" -thin i386 -output "$BINARY_i386"`;

# Retain the orignal binary for QA and use with the util 'atos'
`mv -f "$BINARY" "$BINARY_FULL"`;

# Perform desired stripping on each thin binary.
`strip -S -x -o "${BINARY_ppc}_tmp"  -r "$BINARY_ppc" `;
`strip -S -x -o "${BINARY_i386}_tmp" -r "$BINARY_i386"`;

# We're now done with the original thin binaries, so chuck them.
`rm -f "$BINARY_ppc" `;
`rm -f "$BINARY_i386"`;

# Make the new universal binary from our stripped thin pieces.
`lipo -arch i386 "${BINARY_i386}_tmp" -arch ppc "${BINARY_ppc}_tmp" -create -output "$BINARY"`;

# We're now done with the temp thin binaries, so chuck them.
`rm -f "${BINARY_ppc}_tmp" `;
`rm -f "${BINARY_i386}_tmp"`;


print "\n==================== External strip phase complete\n";


#EOF
```

Follow-Ups:
Re: Symbol stripping tips + handy script
From: Greg Hurrell <email@hidden>

Re: Symbol stripping question
From: Hugh Sontag <email@hidden>

Re: Symbol stripping tips + handy script
From: Rush Manbert <email@hidden>

Prev by Date: **Re: RezMerger size limitation .r --> .rsrc**
Next by Date: **Re: Symbol stripping tips + handy script**
Previous by thread: **Issue with GDB printing long doubles (128 bit)**
Next by thread: **Re: Symbol stripping tips + handy script**
Index(es):
Date
Thread

Home Archives Terms/Conditions Contact RSS Lists About