Single-File / Multi-OS / Multi-Architecture Publishing in .NET

Walkthrough

Assumes .NET 5 or 6. Earlier versions of .NET Core support single file publishing as well, but it's markedly improved in v5 and v6.

Single file publishing is controlled by three arguments to the dotnet publish command:

Argument	Description
-r <rid>	Runtime Identifier. Specifies which operating system, operating system version, and architecture to target.
–self-contained true	Include all dependencies. This removes the need to have the associated .NET runtime installed on the target machine.
/p:PublishSingleFile=true	Bundle the output into a single file.

For example, the command for publishing to a single file, targeting Windows 10 (64-bit), looks like this:

```
dotnet publish -c Release -r win10-x64 --self-contained true \
  /p:PublishSingleFile=true
```

Targeting other OS/architecture combinations is as simple as replacing the RID value in the command with the RID you want to target.

I've tested the following RIDs:

RID	Operating System	Architecture
linux-x64	Linux	64-bit AMD
linux-arm	Linux	32-bit ARM (This can be used to target Raspberry Pi)
osx.10.14-x64	macOS 10.14 Mojave	64-bit ARM
win10-x64	Windows 10	64-bit AMD

You can find more information about RIDs in the .NET Core RID Catalog article.

Results for .NET 5

Technically speaking, the publish command does not produce a "single file" in all cases. Starting with a simple console project named "SingleFilePublish", the following outputs are produced:

Linux

One executable:

```
SingleFilePublish
```

Windows

5 files, including the executable:

```
clrcompression.dll
clrjit.dll
coreclr.dll
mscordaccore.dll
SingleFilePublish.exe
```

macOS

8 files, including the executable:

```
libclrjit.dylib
libcoreclr.dylib
libSystem.IO.Compression.Native.dylib
libSystem.Native.dylib
libSystem.Net.Security.Native.dylib
libSystem.Security.Cryptography.Native.Apple.dylib
libSystem.Security.Cryptography.Native.OpenSsl.dylib
SingleFilePublish
```

Raspberry Pi (32-bit ARM)

One executable:

```
SingleFilePublish
```

Results for .NET 6

With .NET 6, it looks like true single-file publishing has been achieved.

Linux

One executable:

```
SingleFilePublish
```

Windows

One executable:

```
SingleFilePublish.exe
```

macOS

One executable:

```
SingleFilePublish
```

Raspberry Pi (32-bit ARM)

One executable:

```
SingleFilePublish
```

Deployment

When you run the **dotnet publish** command, outputs are written to the **bin/Release/net5.0/publish** directory. To deploy, simply copy the contents of this directory to your target machine.

The publish directory includes a .pdb file, but it's not required. It just contains debugging information.

Sample Makefile

Simplifies the steps described above.

Makefile

```
LINUX_RID = linux-x64
LINUX_RID_ARM = linux-arm
MAC_RID = osx.10.14-x64
WINDOWS_RID = win10-x64
CONFIGURATION_ARGS = Release
PUBLISH_ARGS = --self-contained true /p:PublishSingleFile=true

default:
```

```
    @echo 'Targets:'
    @echo '  run'
    @echo '  run-win'
    @echo '  copy-pi'
    @echo '  publish'
    @echo '  publish-win'
    @echo '  publish-mac'
    @echo '  publish-arm'
    @echo '  clean'

run: publish
    ./bin/Release/net5.0/linux-x64/publish/SingleFilePublish

run-win: publish-win
    wine ./bin/Release/net5.0/win10-x64/publish/SingleFilePublish

copy-pi: publish-arm
    scp ./bin/Release/net5.0/linux-arm/publish/SingleFilePublish
pi@raspi4-main:/home/pi

publish:
    dotnet publish -c $(CONFIGURATION_ARGS) -r $(LINUX_RID)
$(PUBLISH_ARGS)

publish-win:
    dotnet publish -c $(CONFIGURATION_ARGS) -r $(WINDOWS_RID)
$(PUBLISH_ARGS)

publish-arm:
    dotnet publish -c $(CONFIGURATION_ARGS) -r $(LINUX_RID_ARM)
$(PUBLISH_ARGS)

publish-mac:
    dotnet publish -c $(CONFIGURATION_ARGS) -r $(MAC_RID)
$(PUBLISH_ARGS)

clean:
    -rm -rf bin/
    -rm -rf obj/
```

cross platform, dotnet