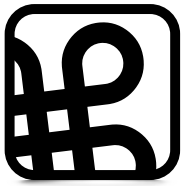


Macoun

iOS-Frameworks selber bauen

Clemens Wagner
macmoonshine@gmx.de

Einleitung

Motivation

Code wiederverwenden

- Sourcecode
- Objektcode ⇒ Bibliothek oder Framework

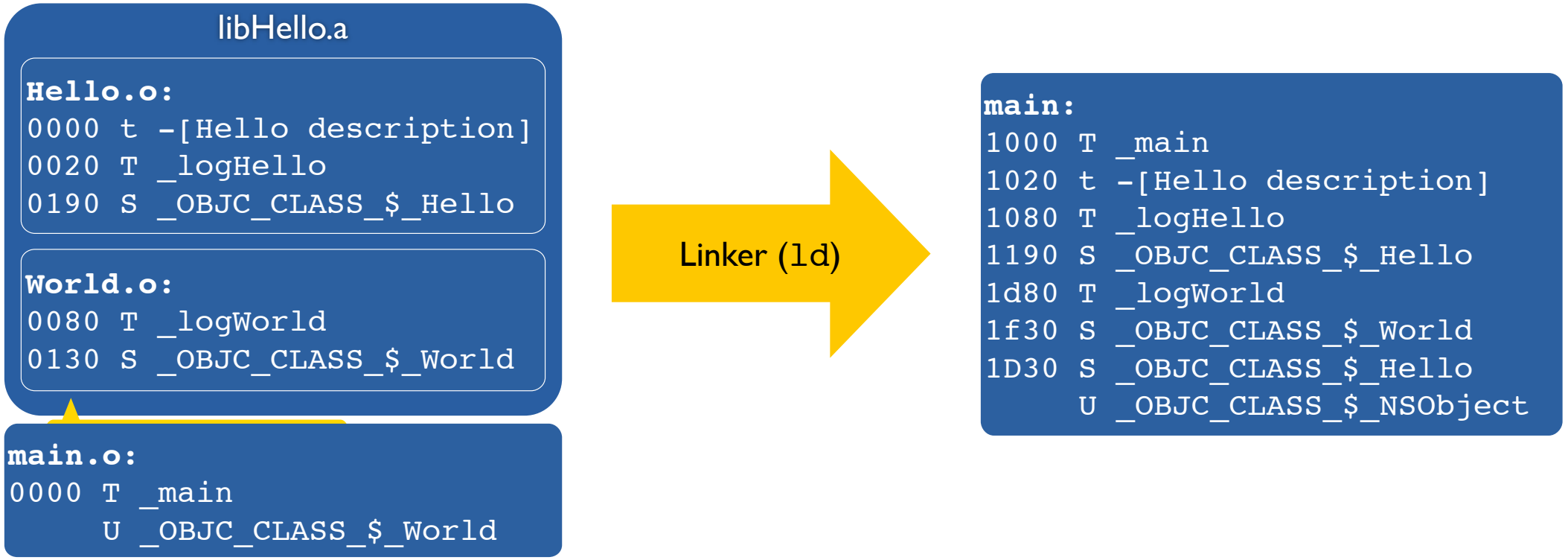
Wiederverwendung Objektcode

- schnellerer Buildprozess
- Platzersparnis
- (Open-Source-)Libs: Konfiguration und Einbindung
- Produkt: Implementierung verstecken

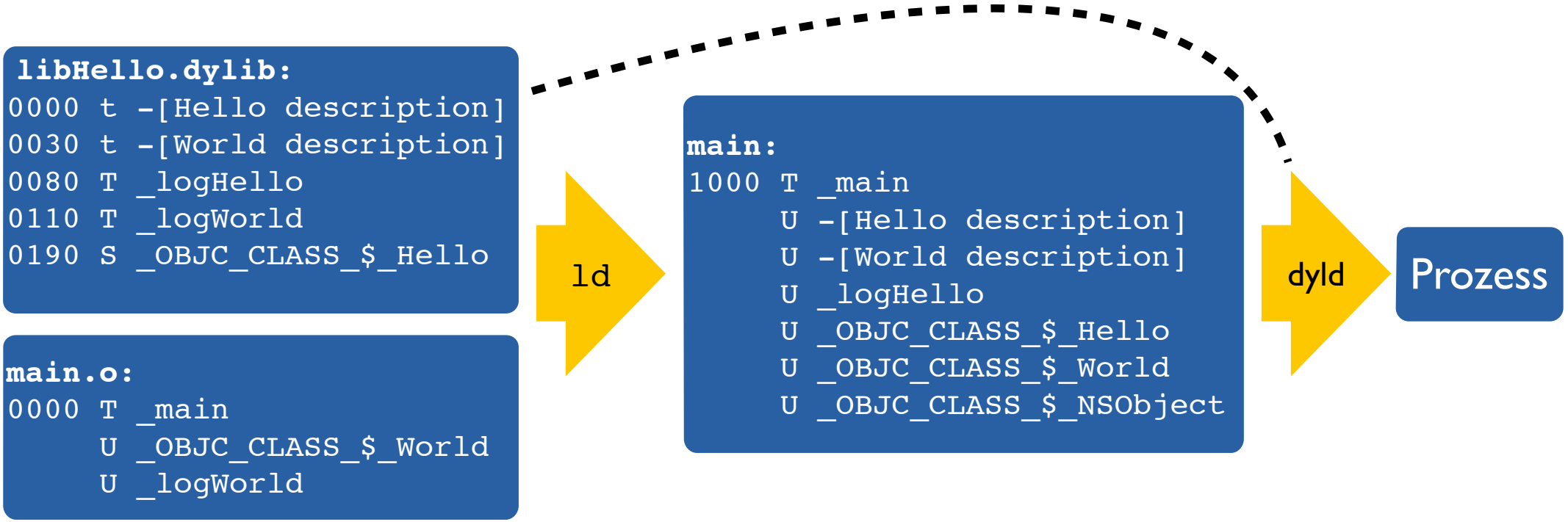
Bibliotheken

- Klassen
- Funktionen
- Konstanten
- statisch oder dynamisch

Statische Bibliothek



Dynamische Bibliothek



Dynamische Bibliotheken anzeigen

```
otool -L main
main:
  libHello.dylib (compatibility version 0.0.0, current version 0.0.0)
  /System/Library/Frameworks/Foundation.framework/Versions/C/Foundation \
    (compatibility version 300.0.0, current version 1056.13.0)
  /usr/lib/libSystem.B.dylib (compatibility version 1.0.0, \
    current version 1197.1.1)
  /System/Library/Frameworks/CoreFoundation.framework/Versions/A/CoreFoundation \
    (compatibility version 150.0.0, current version 855.14.0)
  /usr/lib/libobjc.A.dylib (compatibility version 1.0.0, current version 228.0.0)
```

Vorteile dynamische Bibliothek

- Platzsparend
  + Festplatte
  + Hauptspeicher
- Aktualisierbar

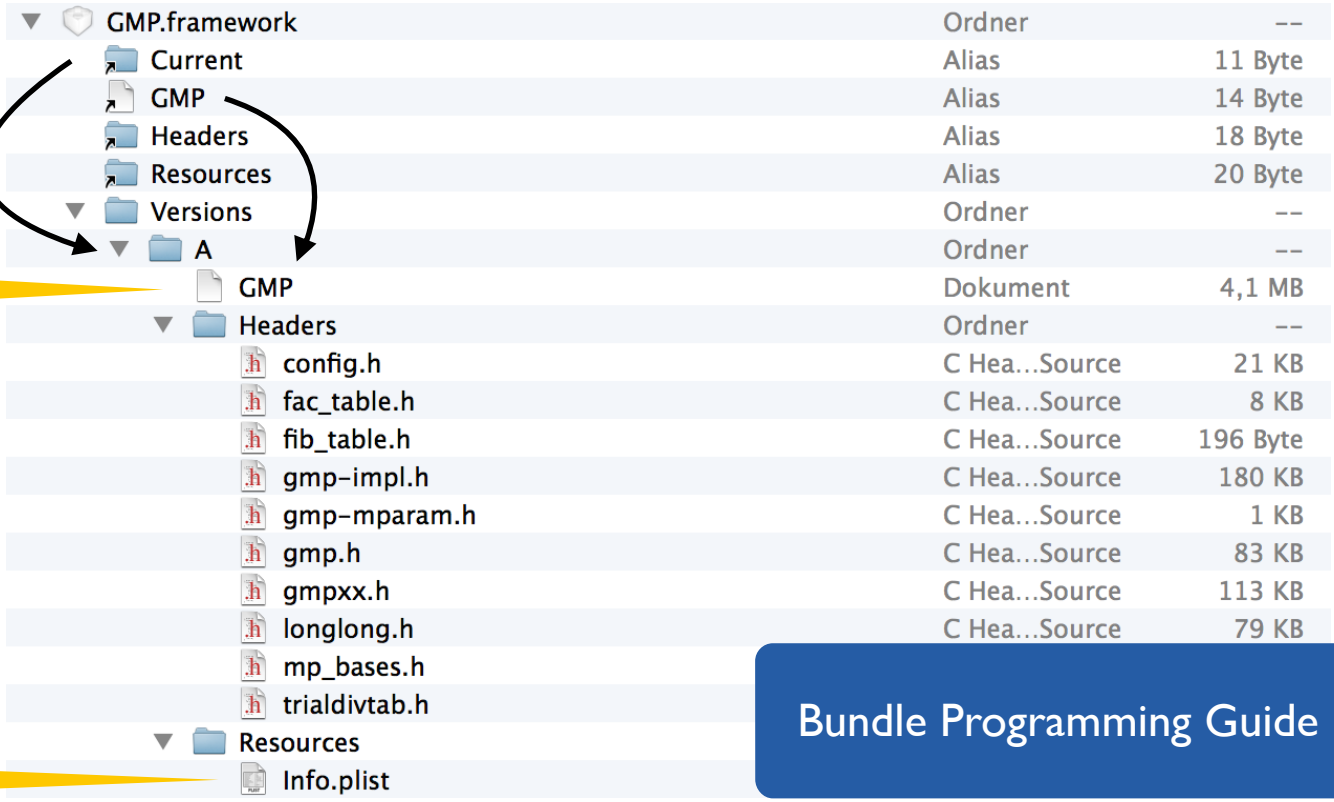
Dynamische Bibliotheken unter iOS

- Systembibliotheken √
- Nutzerbibliotheken (mehrere Apps? / Installationsprozess?)
- iOS 8: App-Erweiterungen √

Frameworks

- Bibliothek
- Headerdateien
- ~~Ressourcen~~ (statisch)

Struktur von Frameworks



Bibliothek

`CFBundlePackageType = FMWK`

Bundle Programming Guide

Allgemeinb(u)ildung

Statische Bibliothek bauen

- Übersetzen
- Archivieren
- Indizieren

```
$ clang -c Hello.m World.m
$ ar q libHello.a Hello.o World.o
$ ranlib libHello.a
```

Statische Bibliothek bauen

- Übersetzen
- Archivieren
- Indizieren

```
$ clang -c Hello.m World.m
$ libtool -static -o libHello.a Hello.o World.o
```

Dynamische Bibliothek bauen

- Übersetzen
- Binden

```
$ clang -c Hello.m World.m
$ clang -dynamiclib -o libHello.dylib \
   Hello.o World.o -framework Foundation
```

Dynamische Bibliothek bauen

- Übersetzen
- Binden

```
$ clang -c Hello.m World.m
$ libtool -dynamic -o libHello.dylib \
   Hello.o World.o -framework Foundation
```

Buildprozess skripten

- Makefile
- Variablen, Abhängigkeiten und Regeln
- Standardvariablen und -regeln
  /Applications/Xcode.app/Contents/Developer/Makefiles/

Makefile: Variablen

```
CC = clang
LD = clang
LDFLAGS = $(CFLAGS)
RANLIB = ranlib -s
SRCS = Hello.m World.m
OBJS = $(patsubst %.m, %.o, $(SRCS))
```

OBJS = Hello.o World.o

Makefile: Abhängigkeiten

```
all: static dynamic

static: libHello.a
dynamic: libHello.dylib
```

Makefile: Regeln

```
%.a: $(OBJS)
    $(AR) q $@ $^
    $(RANLIB) $@

%.dylib: $(OBJS)
    $(LD) $(LDFLAGS) -dynamiclib -framework Foundation $^ -o $@

clean:
___$(RM) *.o *.a *.dylib *.so
```

Ich bin ein Tab.

make

```
$ make
clang -std=c99 -O2   -c -o Hello.o Hello.m
clang -std=c99 -O2   -c -o World.o World.m
ar q libHello.a Hello.o World.o
ar: creating archive libHello.a
ranlib -s libHello.a
clang -std=c99 -O2 -dynamiclib -framework Foundation \
   Hello.o World.o -o libHello.dylib
rm World.o Hello.o
```

make love

```
$ make static
clang -std=c99 -O2   -c -o Hello.o Hello.m
clang -std=c99 -O2   -c -o World.o World.m
ar q libHello.a Hello.o World.o
ar: creating archive libHello.a
ranlib -s libHello.a
rm World.o Hello.o
```

Variablen überschreiben

```
$ make CC=gcc dynamic LD=gcc
gcc -std=c99 -O2 -c -o Hello.o Hello.m
gcc -std=c99 -O2 -c -o World.o World.m
gcc -std=c99 -O2 -dynamiclib -framework Foundation \
  Hello.o World.o -o libHello.dylib
rm World.o Hello.o
```

... und was ist mit dem iPhone?

Cross Compiling

Plattformen

- iPhoneSimulator
- iPhoneOS
- (MacOSX)

SDKs

- Beschreibungen (Architekturen, Werkzeuge)
- spezifische Erweiterungen / Tools
- Frameworks, Bibliotheken und Header

Prozessor-Architekturen

- MacOSX / iPhoneSimulator:

  i386, x86_64

- iPhoneOS:

  armv7, armv7s, arm64

Cross-Compiling

- MacOSX
- x86_64

```
$ clang -c Hello.m
```

SDK wählen

- iphoneos
- x86_64

```
$ xcrun --sdk iphoneos clang -c Hello.m
```

Architektur wählen

- iphoneos
- armv7

```
$ xcrun --sdk iphoneos clang -arch armv7 -c Hello.m
```

Wo laufen sie denn?

- iOS Deployment Target
- Für alle Plattformen und SDKs
  `-miphoneos-version-min=6.0`

```
$ otool -l main | fgrep -A 3 -B 1 LC_VERSION_MIN
Load command 8
      cmd LC_VERSION_MIN_IPHONEOS
  cmdsize 16
  version 6.0
      sdk 7.1
```

Buildprozess anpassen ...

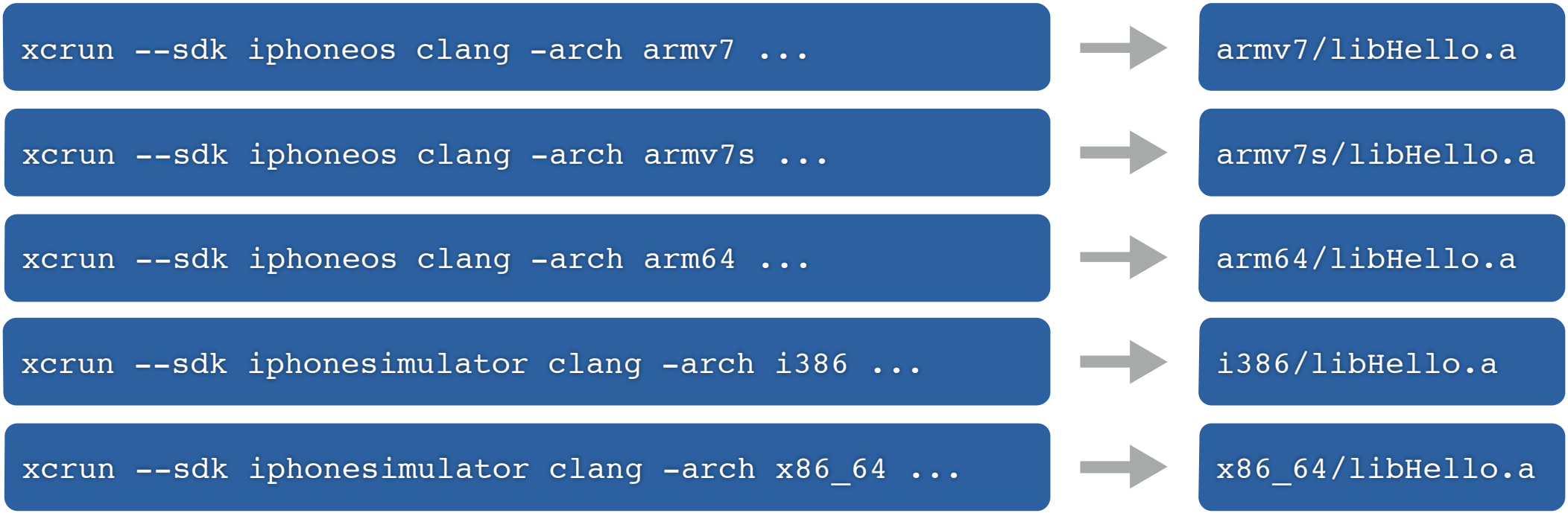
```
SDK = macosx
XCRUN = xcrun --sdk $(SDK)
CC = $(XCRUN) clang
LD = $(XCRUN) clang
LDFLAGS = $(CFLAGS)
[...]
```

… und ausführen

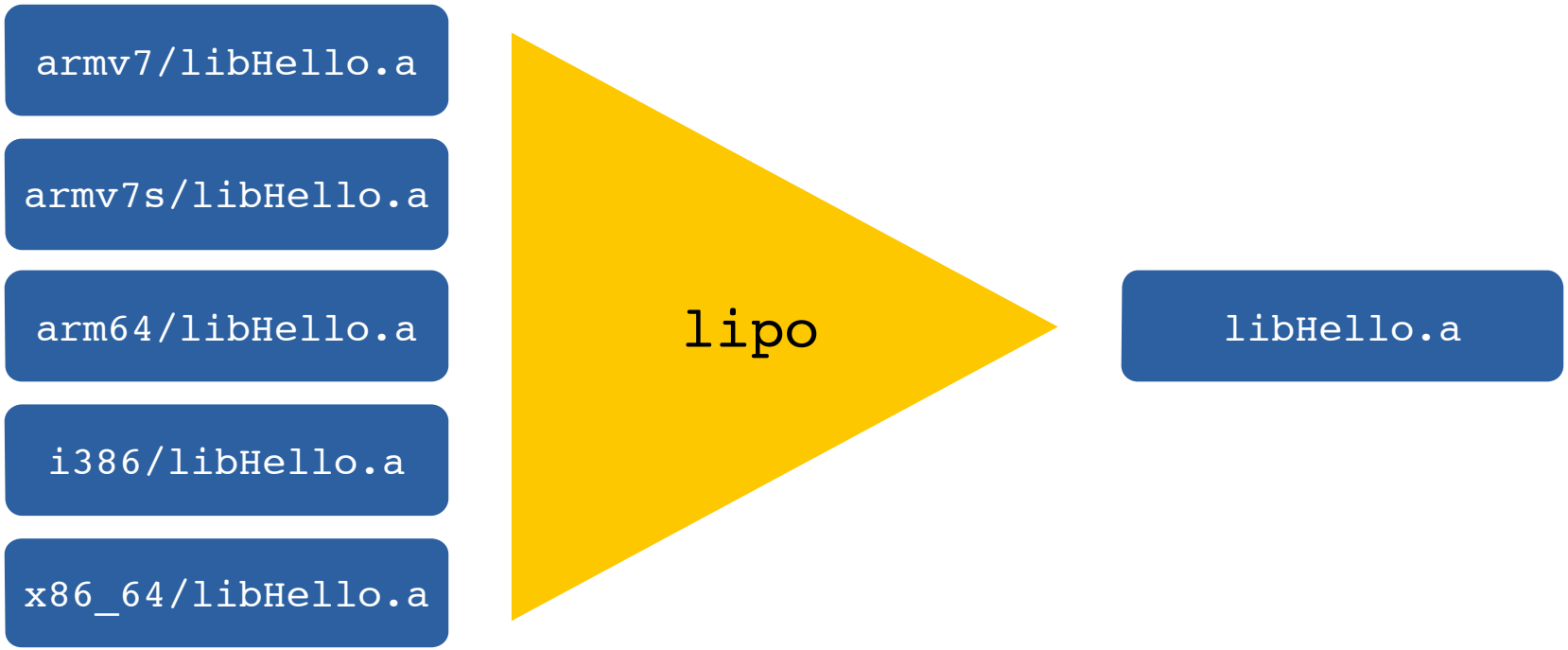
```
arch=armv7
sdk=iphoneos
build_dir="$BUILD_DIR/$arch"
mkdir -p $build_dir
make CFLAGS="-arch $arch -miphoneos-version-min=6.0" SDK=$sdk \
  $build_dir/libHello.a
```

Universal Binaries

5x bauen = 5 Libs?



Universal Binary: Aus 5 mach 1



Lipo

```
lipo -arch armv7 armv7/libHello.a \
  -arch armv7s armv7s/libHello.a \
  -arch arm64 arm64/libHello.a \
  -arch i386 i386/libHello.a \
  -arch x86_64 x86_64/libHello.a \
  -create -output libHello.a
```

Demo

Statische Bibliothek einbinden

- Bibliothek zu „Linked Frameworks and Libraries“
- Laden erzwingen:
  - `LDFLAGS = -ObjC` // alle Objective-C-Klassen
  - `LDFLAGS = -all_load` // alle Symbole
  - `LDFLAGS = -force_load` *Lib* // alle Symbole aus Lib

Dynamische Bibliothek einbinden (I)

- Bibliothek zu „Linked Frameworks and Libraries“
- Bibliothek ins Bundle kopieren
- Pfad korrigieren

Dynamische Bibliothek einbinden (II)

- Pfad korrigieren

```
for arch in $ARCHS
do
    install_name_tool -change build/$arch/libHello.dylib \
    '@executable_path/libHello.dylib' \
    $BUILT_PRODUCTS_DIR/$EXECUTABLE_PATH
done
```

Dynamische Bibliothek laden

```
NSString *thePath = [[NSBundle mainBundle] pathForResource:@"libHello" \
  ofType:@"dylib"];
void *theLib = dlopen([thePath fileSystemRepresentation], \
  RTLD_LAZY | RTLD_LOCAL | RTLD_FIRST);

if(theLib == NULL) {
  NSLog(@"dlerror: %s", dlerror());
}
```

Symbole laden

- Klassen √
- Funktionen als Pointer deklarieren
- Symbol über `dlsym` binden

```
void (*logWorld)() = NULL;

logWorld = dlsym(theLib, "logWorld");
```

Dynamische Bibliothek und Code Signing (I)

```
Unable to get entitlements for client task.
Error: Error Domain=NSPOSIXErrorDomain Code=-1
"The operation couldn’t be completed. (POSIX
error -1 - Unknown error: -1)"
```

Dynamische Bibliothek und Code Signing (II)

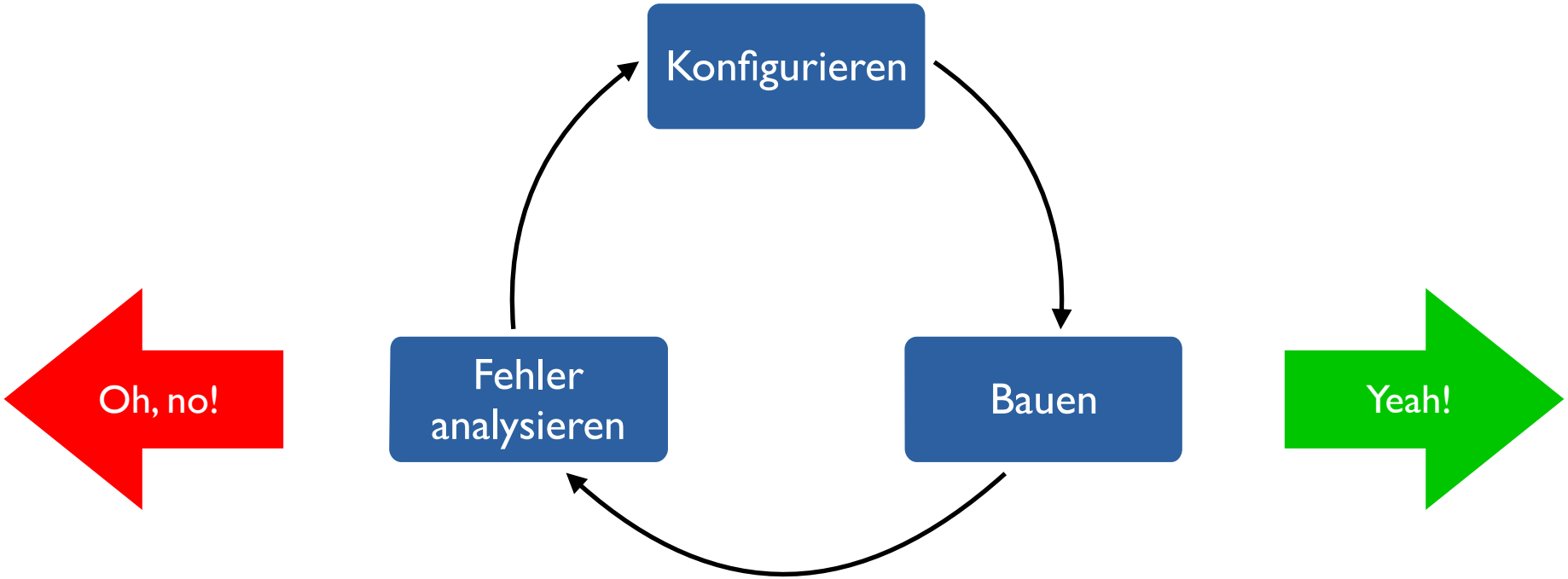
```
$ codesign --sign "iPhone Developer" libHello.dylib
$ otool -l libHello.dylib | fgrep -A 3 -B 1 LC_CODE_SIGNATURE
Load command 15
      cmd LC_CODE_SIGNATURE
  cmdsize 16
  dataoff 9376
 datasize 9392
```

Fallbeispiele

Open-Source-Projekte

- libpq (Postgres)
- Tesseract (OCR)
- gmp (beliebig genaue Zahlen)

Buildprozess anpassen



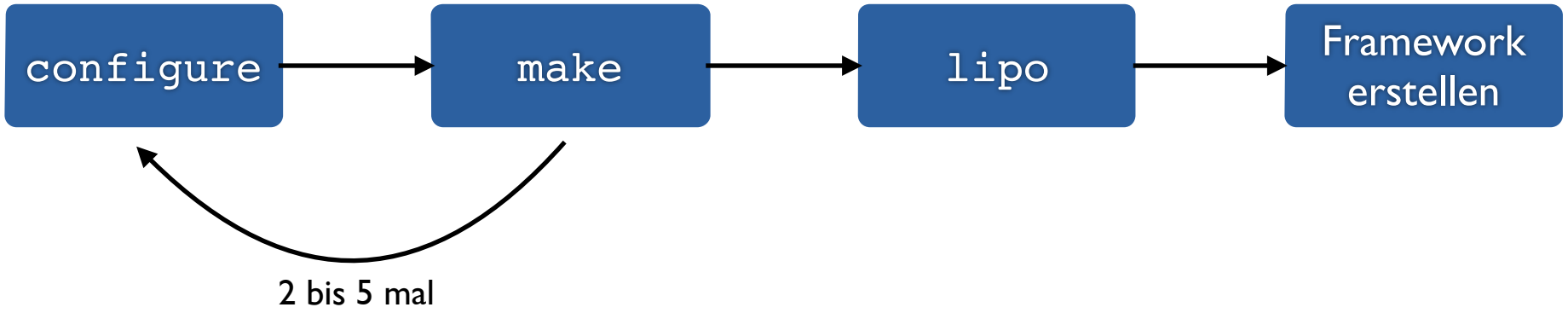
Konfiguration

- Aus- und Einschalten von Optionen (`configure`, `cmake`)
- Compiler über `CFLAGS`
- Linker über `LDFLAGS`
- ggf. weitere Möglichkeiten

Fehlersuche

- Fehlermeldungen in `config.log`
- Achtung! Libs in lokalen Verzeichnissen (z. B. `/opt/local/lib`)

Bauen



Demo

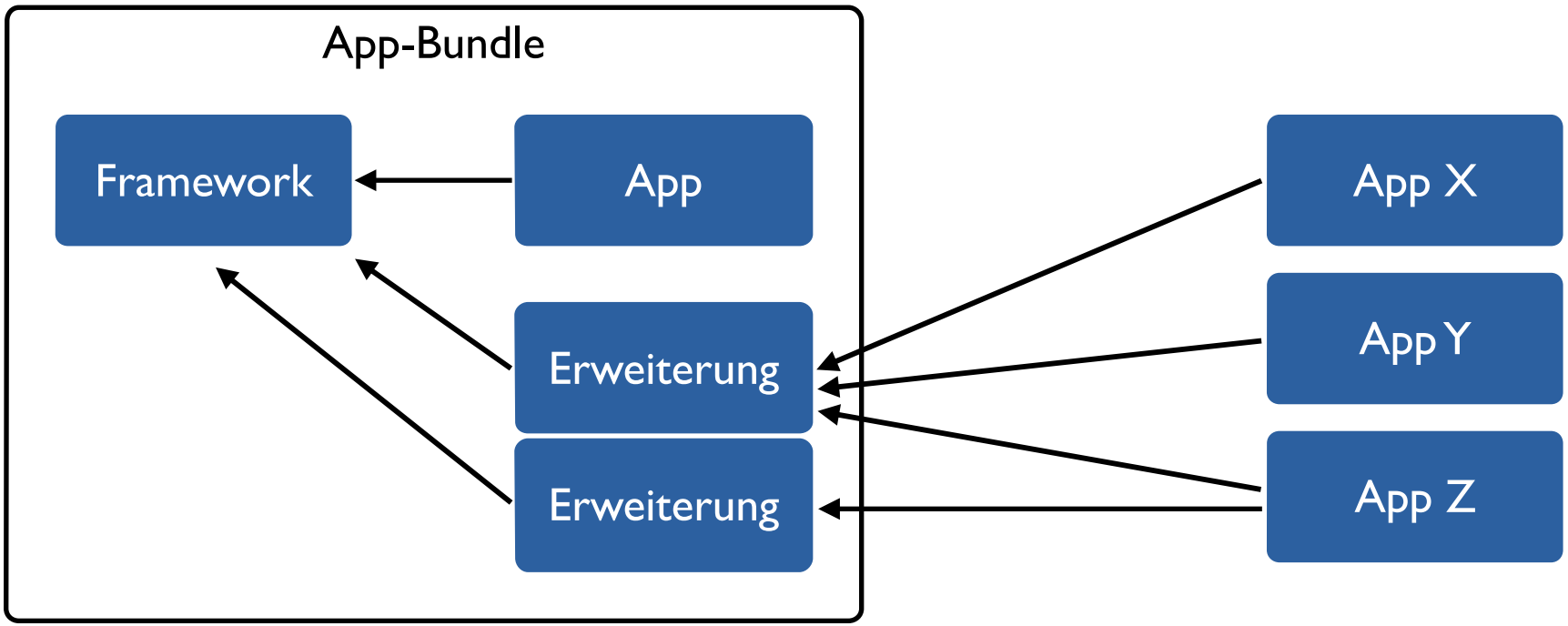
Frameworks unter iOS 8

Dynamische Frameworks

- Warum jetzt doch?
- innerhalb eines App-Bundles
- ARM64-Support für App-Store

App Extension
Programming Guide

Erweiterungen



Kompatibilität

- bis iOS 7.1: keine dynamischen Frameworks erlaubt!
- kein Weak-Linking
- Framework unter iOS 8 dynamisch nachladen

Dynamisch nachladen

```
if([[[UIDevice currentDevice] systemVersion] floatValue] >= 8.0f) {




}
```

Dynamisch nachladen

```
if([[[UIDevice currentDevice] systemVersion] floatValue] >= 8.0f) {
    NSBundle *theMainBundle = [NSBundle mainBundle];
    NSURL *theURL = [theMainBundle URLForResource:@"MyFramework" \
        withExtension:@"framework" subdirectory:@"Frameworks"];



}
```

Dynamisch nachladen

```
if([[[UIDevice currentDevice] systemVersion] floatValue] >= 8.0f) {
    NSBundle *theMainBundle = [NSBundle mainBundle];
    NSURL *theURL = [theMainBundle URLForResource:@"MyFramework" \
        withExtension:@"framework" subdirectory:@"Frameworks"];
    NSBundle *theBundle = [NSBundle bundleWithURL:theURL];

    [theBundle load];
}
```

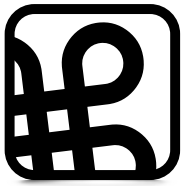
Zusammenfassung

- statische Frameworks
  ⇒ Open-Source-Code / eigene Libs

- dynamische Frameworks
  ⇒ App-Extensions

Fragen?

Vielen Dank



Macoun