Bonus Lecture 1

Using Libraries and Distributing Your Software

Anton Gerdelan <gerdela@scss.tcd.ie>

Installing Libraries

- `sudo apt-get install` … , Synaptic
- `brew install` …
- download pre-compiled binaries from website
  - must match your
    - compiler (C++) or version of Visual Studio
    - 32-bit or 64-bit build
    - release or debug build
- download the source code and build them yourself

Compiling Libraries

- If the library is a single .h -> no need!
- Read the instructions - does the library use
  - make - we know this one!
  - automake - slightly more complex - checks for dependencies
  - cmake - download cmake
  - qtmake - probably not a good sign
  - something else…

What Builds?

- Linux
  - .so - "shared object" - **dynamic library**
  - .a - "archive" - **static library**
- Mac
  - .dylib - **dynamic library**
  - .a - **static library**
  - .framework - a **package** containing headers and libraries
- Windows
  - .dll - "dynamically linked library"- **dynamic library**
  - .lib - usually a **stub** that goes with the .dll, may also be a **static** library
  - .a - some compilers (not Visual Studio) - **static library**

Static Libraries

- static libraries work like compiled .c source
  - `gcc -o demo main.c` **`somelibrary.a`**
- usually depend on other libraries
- `gcc -o demo main.c` **`somelibrary.a -lotherlibrary`**
- read the instructions to find out - or guess from linker complaints
- compiles into your binary program - <u>easier to distribute</u>

Dynamic Libraries

- newer. a pain. "advantages": re-use and upgrade
- add path to library file (-L with gcc) if not on system path
- add file to link (-l with gcc)
    - if file is called **libopengl32.so** then just **-lopengl32**
- `gcc -o demo main.c` **`-Lmy_libs_folder -lmylib`**
- does not get compiled into binary program
- has to stay on system path
    - or as a loose file that you include with your program*****

Headers

- Libraries usually also ship with headers (.h or .hpp)
  - is there an `include/` folder in the download?
- Copy this into your project (unless it's installed on system path)
- Tell compiler where to find this folder too
  - with gcc with is -I (capital i)
    - `gcc -o myprogram main.c` **`-Iincludes/`** `-Llibs/ -lmylib.so`

How to Distribute with Dynamic Libraries

- Find all the dependencies, then:

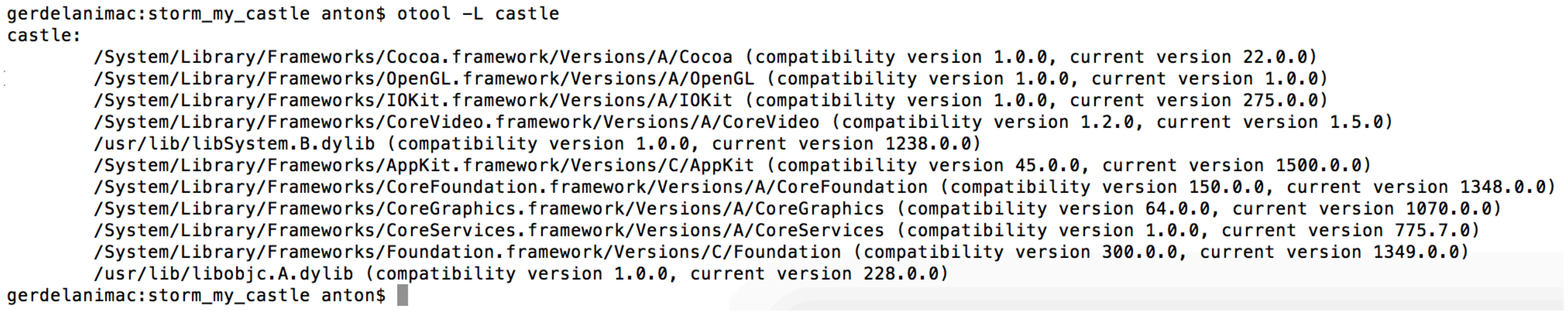
1. if it looks like a system library - safely ignore
2. tell users to install first (depends on licences/project) or
3. provide redistributable (e.g. DirectX 2008 redist) or
4. make it an automatically installed dependency (Linux)

$\longrightarrow$ 5. compile it in as a **static library** or

6. remove it from your project and **write your own** or
7. **if all else fails** - include the dynamic library in your bundle

How to Query Dependencies

- Linux: use **`ldd my_program`**
  - then ldd on each dependency within
- Mac: use **`otool -L my_program`**
- Windows: download **Dependency Walker**
  - tree view of dependencies

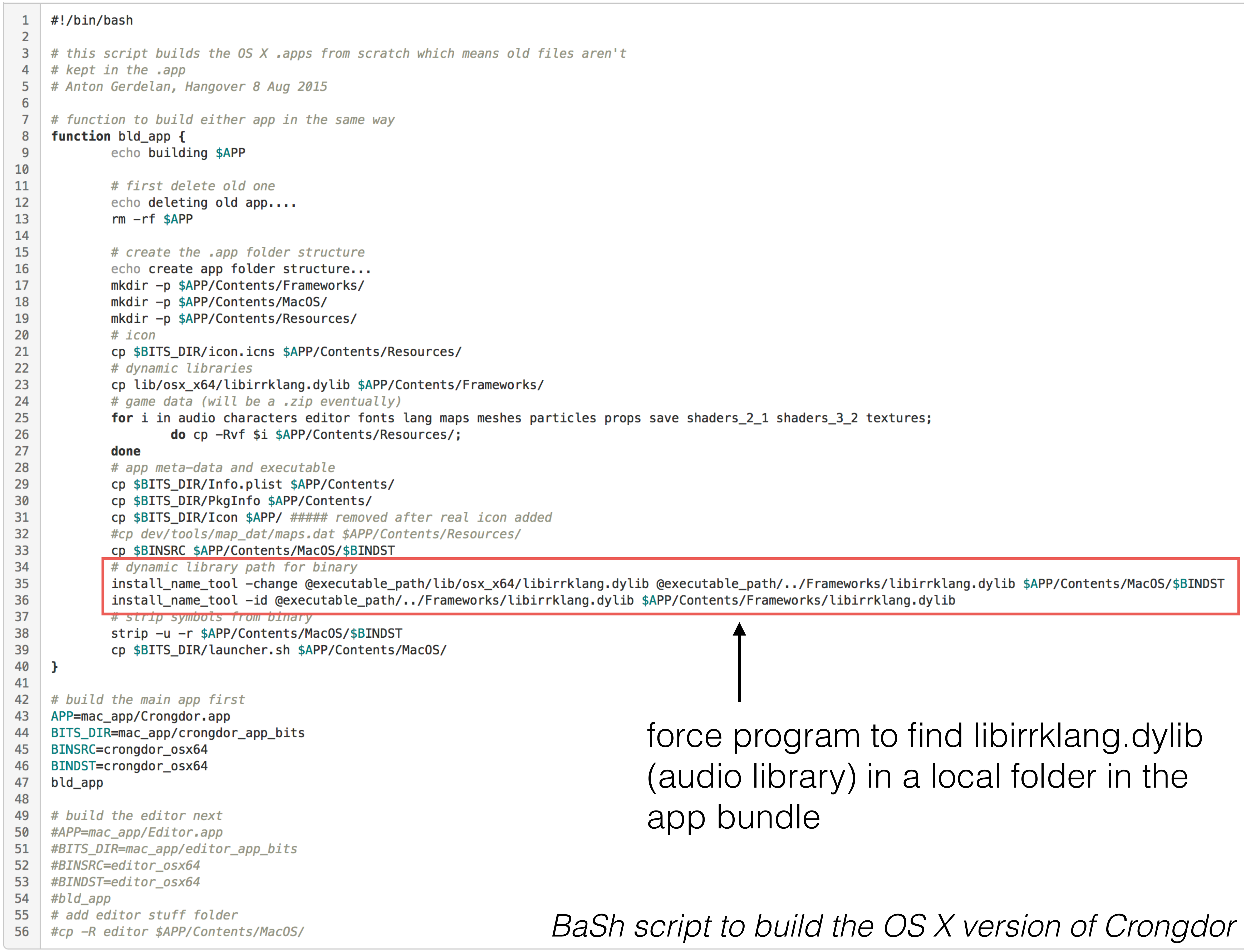


```
gerdelanimac:storm_my_castle anton$ otool -L castle
castle:
	/System/Library/Frameworks/Cocoa.framework/Versions/A/Cocoa (compatibility version 1.0.0, current version 22.0.0)
	/System/Library/Frameworks/OpenGL.framework/Versions/A/OpenGL (compatibility version 1.0.0, current version 1.0.0)
	/System/Library/Frameworks/IOKit.framework/Versions/A/IOKit (compatibility version 1.0.0, current version 275.0.0)
	/System/Library/Frameworks/CoreVideo.framework/Versions/A/CoreVideo (compatibility version 1.2.0, current version 1.5.0)
	/usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current version 1238.0.0)
	/System/Library/Frameworks/AppKit.framework/Versions/C/AppKit (compatibility version 45.0.0, current version 1500.0.0)
	/System/Library/Frameworks/CoreFoundation.framework/Versions/A/CoreFoundation (compatibility version 150.0.0, current version 1348.0.0)
	/System/Library/Frameworks/CoreGraphics.framework/Versions/A/CoreGraphics (compatibility version 64.0.0, current version 1070.0.0)
	/System/Library/Frameworks/CoreServices.framework/Versions/A/CoreServices (compatibility version 1.0.0, current version 775.7.0)
	/System/Library/Frameworks/Foundation.framework/Versions/C/Foundation (compatibility version 300.0.0, current version 1349.0.0)
	/usr/lib/libobjc.A.dylib (compatibility version 1.0.0, current version 228.0.0)
gerdelanimac:storm_my_castle anton$ 
```

I think these are all OS X system libraries

How to Distribute with Dynamic Libraries

- Windows - put the .dll files into your program's folder
    - other OSs don't allow this - security vulnerability
- Linux
    - enter into console before compiling:
    - `export LD_RUN_PATH=my_libs_folder/`
- Mac
    - ~put into a **.app bundle** folder structure
    - use **install_name_tool** on libraries and program binary
- I make scripts to do all of this ugly stuff



```bash
#!/bin/bash

# this script builds the OS X .apps from scratch which means old files aren't
# kept in the .app
# Anton Gerdelan, Hangover 8 Aug 2015

# function to build either app in the same way
function bld_app {
        echo building $APP

        # first delete old one
        echo deleting old app....
        rm -rf $APP

        # create the .app folder structure
        echo create app folder structure...
        mkdir -p $APP/Contents/Frameworks/
        mkdir -p $APP/Contents/MacOS/
        mkdir -p $APP/Contents/Resources/
        # icon
        cp $BITS_DIR/icon.icns $APP/Contents/Resources/
        # dynamic libraries
        cp lib/osx_x64/libirrklang.dylib $APP/Contents/Frameworks/
        # game data (will be a .zip eventually)
        for i in audio characters editor fonts lang maps meshes particles props save shaders_2_1 shaders_3_2 textures;
                do cp -Rvf $i $APP/Contents/Resources/;
        done
        # app meta-data and executable
        cp $BITS_DIR/Info.plist $APP/Contents/
        cp $BITS_DIR/PkgInfo $APP/Contents/
        cp $BITS_DIR/Icon $APP/ ##### removed after real icon added
        #cp dev/tools/map_dat/maps.dat $APP/Contents/Resources/
        cp $BINSRC $APP/Contents/MacOS/$BINDST
        # dynamic library path for binary
        install_name_tool -change @executable_path/lib/osx_x64/libirrklang.dylib @executable_path/../Frameworks/libirrklang.dylib $APP/Contents/MacOS/$BINDST
        install_name_tool -id @executable_path/../Frameworks/libirrklang.dylib $APP/Contents/Frameworks/libirrklang.dylib
        # strip symbols from binary
        strip -u -r $APP/Contents/MacOS/$BINDST
        cp $BITS_DIR/launcher.sh $APP/Contents/MacOS/
}

# build the main app first
APP=mac_app/Crongdor.app
BITS_DIR=mac_app/crongdor_app_bits
BINSRC=crongdor_osx64
BINDST=crongdor_osx64
bld_app

# build the editor next
#APP=mac_app/Editor.app
#BITS_DIR=mac_app/editor_app_bits
#BINSRC=editor_osx64
#BINDST=editor_osx64
#bld_app
# add editor stuff folder
#cp -R editor $APP/Contents/MacOS/
```

force program to find libirrklang.dylib (audio library) in a local folder in the app bundle

BaSh script to build the OS X version of Crongdor

Can I Make Libraries?

- single-header style, *or*
- have
    - a C file(s) with functions
    - don't have a `main()`
    - header file as interface (declarations of the functions)
- `gcc -o anton.o -c anton.c`
    - might need the `-fPIC` flag above for shared library
- tell the compiler to output library instead
- **`ar`** `rcs` **`libanton.a`** `anton.o`
    - use the archiver to build a static library
- `gcc` **`-shared`** `-o` **`libanton.so`** `anton.o`

Example

- I downloaded the GLFW and GLEW OpenGL helper libraries
  - binaries available for some compilers
  - otherwise require CMake to build from source code
- Download CMake and run it on the project
  - command line tool is a bit clunky
  - use cmake-gui on folder containing cmake files
- This builds a Makefile or VS project file
  - then make that
  - then find in the output libraries and also grab the headers folder