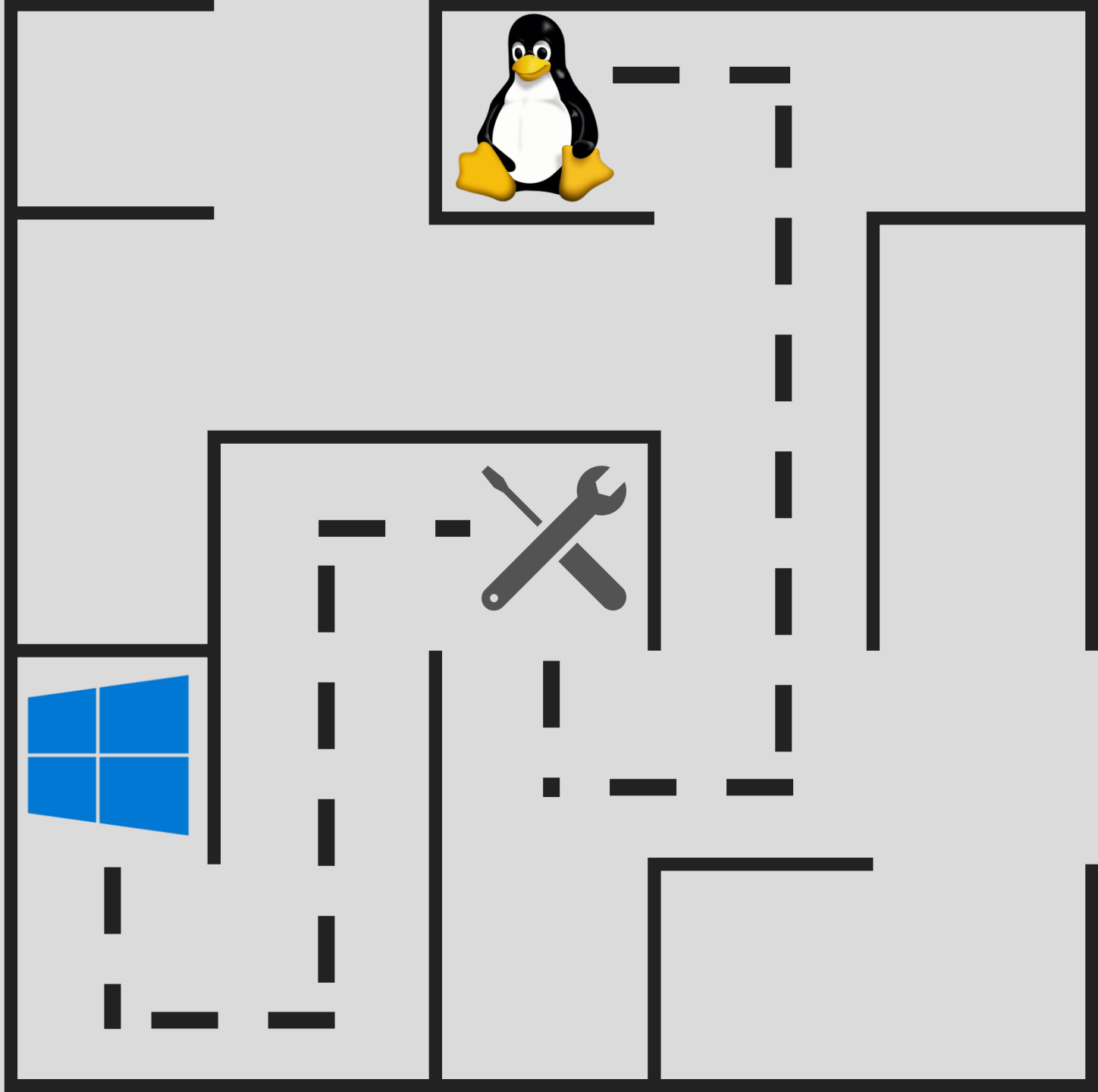
Microsoft

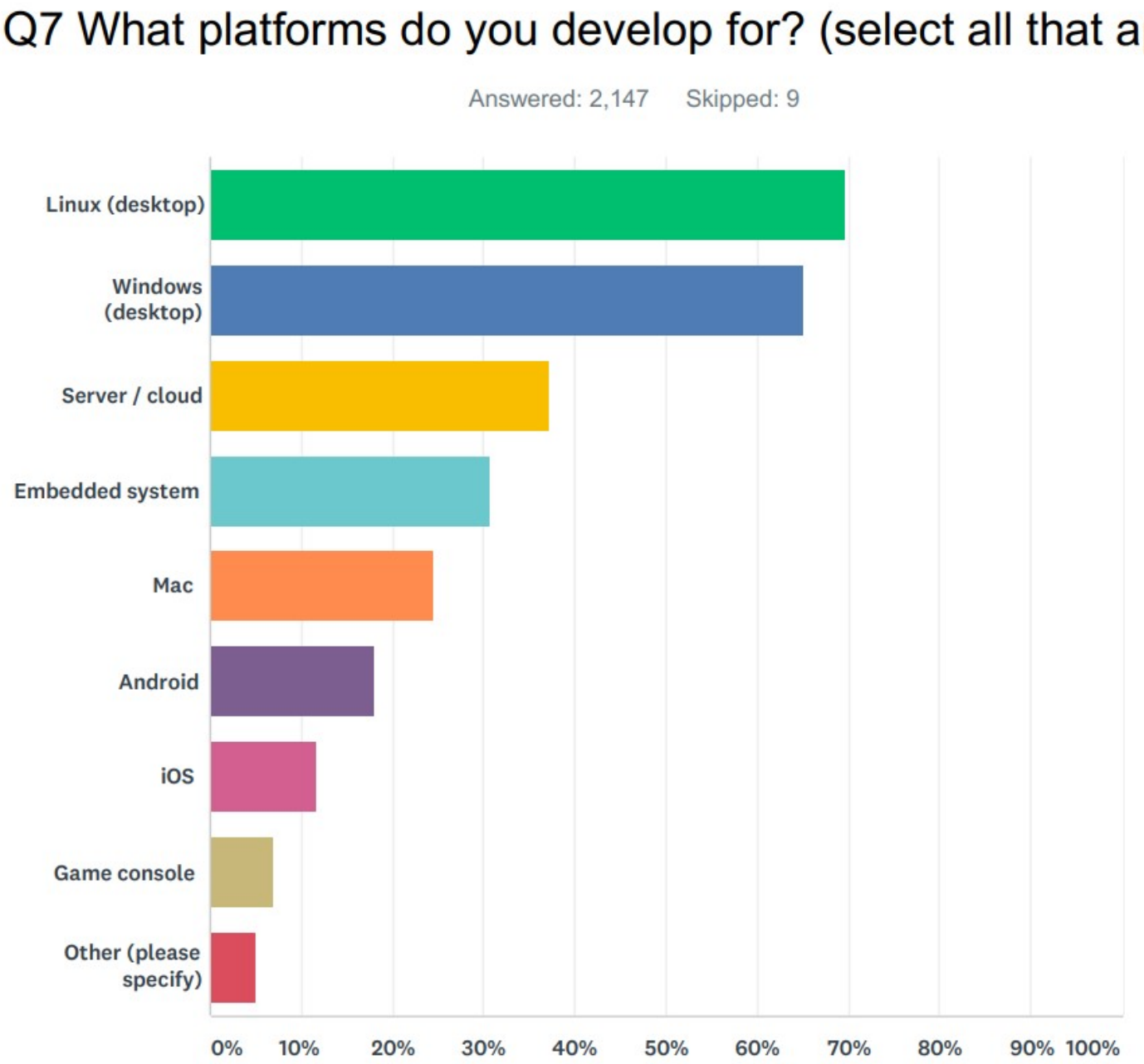
Tools to Ease Cross-platform Development



Elizabeth Morrow
Software Engineer

Augustin Popa
Program Manager

Microsoft C++ product team



Source: ISO C++ 2019 Annual Developer

What do we mean by “cross-platform”?

- Writing code that must compile and run on multiple platforms

- Writing code on one platform but building/running on another. Not Cross Targeting.
  - E.g. Deploying to a server running Linux from a Windows developer machine

Tools covered in this talk

CMake

Windows
Subsystem for Linux

Visual Studio
IDE

Visual Studio
Code

Vcpkg

CMake

- Open source, cross-platform project system/”meta build system”
- Most popular C++ build system for open source projects
- Configured through CMakeLists.txt file
- Generates build files (Unix Makefiles, MSBuild projects, …)
- Cache generation -> Run CMake -> Native build process

- https://cmake.org/

Vcpkg

- Open source C++ library manager (over 1,200 popular libraries)
- Runs on Linux, macOS, and Windows
- Builds from source, saves you time
- Libraries in catalog are ABI compatible with each other
- Integration with Visual Studio and Visual Studio Code

https://github.com/microsoft/vcpkg

Visual Studio Code

- Free, open source multi-language code editor
- Runs on Linux, macOS, and Windows
- Most popular code editor in the world (Source: StackOverflow)
- C++ extension features:
  - Semantic colorization
  - Code completion
  - Live error/warnings
  - Support for your favorite compiler and debugger
  - Productivity features like Rename, Go to Definition, and Find All References
- Native CMake project support (with CMake Tools extension)

- http://code.visualstudio.com/

Demo: Cross-platform development with Visual Studio Code + CMake Tools

Augustin Popa

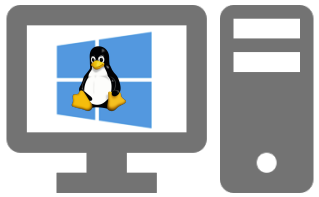
VS Code Demo Summary + Other Cool Features

Covered in demo:

- Create or bring CMake to VS Code with CMake Tools extension
- Edit, build, debug on any OS
- Productivity features like Go to Definition, Rename, Find All References

Other cool features:

- Remotely target a machine, container, or WSL from Visual Studio Code
- Run your dev environment in the cloud with Visual Studio Online
- Multi-root workspaces with CMake Tools



Windows Subsystem for Linux

- Windows 10 feature that provides a native Linux command-line interface
- Commonly used to work with Linux-first ecosystems (Ruby, Python)
- WSL2 announced, running Linux kernel inside light VM
- Use Linux command line tools, or prototype your cross-platform code

- https://docs.microsoft.com/en-us/windows/wsl/install-win10

Visual Studio IDE

- Best in class C++ tools
- Now supports wide range of cross-platform scenarios
- Build/debug on a Linux VM using MSBuild or CMake
- Incremental source synchronization
- Full Visual Studio debugging experience backed by GDB
- First class CMake support including project editing and CMakeLists.txt IntelliSense
- Easy JSON build configuration for multiple platforms
- https://visualstudio.microsoft.com/vs/

Demo: Cross-platform development with Visual Studio 2019

Elizabeth Morrow

Visual Studio Demo Key Points

Covered in demo

- Vcpkg for dependency acquisition
- Leverage Visual Studio C++ productivity tools to write code for Linux
- First class CMake support, including project editing
- Use CMakeSettings configurations to easily switch platforms

- Other tools (not covered):
- Google Test & Boost.Test integration
- clang-tidy and clang-format support
- Clang/LLVM support on Windows & Linux

Key Takeaways

- You can easily build Linux apps even if your dev box is not Linux
- You don’t even need a full Linux machine to get started if you use Visual Studio and target WSL
- CMake is a good choice for projects targeting multiple platforms
- Modern development environments already support CMake
  - Including Visual Studio and Visual Studio Code
- Manage your library dependencies on any OS with vcpkg

Resources

- Download CMake
- Getting started with WSL (docs)
- Download Visual Studio IDE
- Download Visual Studio Code
  - + C++, CMake Tools, Remote Development extensions
- Microsoft C++ team blog

- Codebases used:
  - Open RCT2: https://github.com/OpenRCT2/OpenRCT2
    - Note: requires assets from original game to run
  - Super Tux: https://github.com/WillBuik/supertux (fork of original)