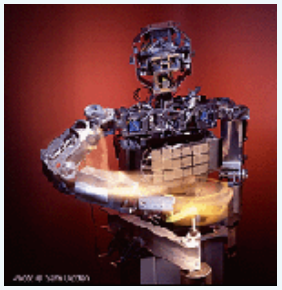
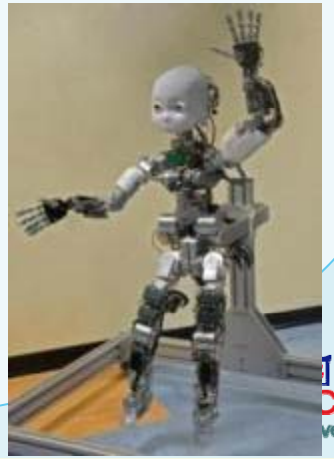
Yet Another Robot Platform

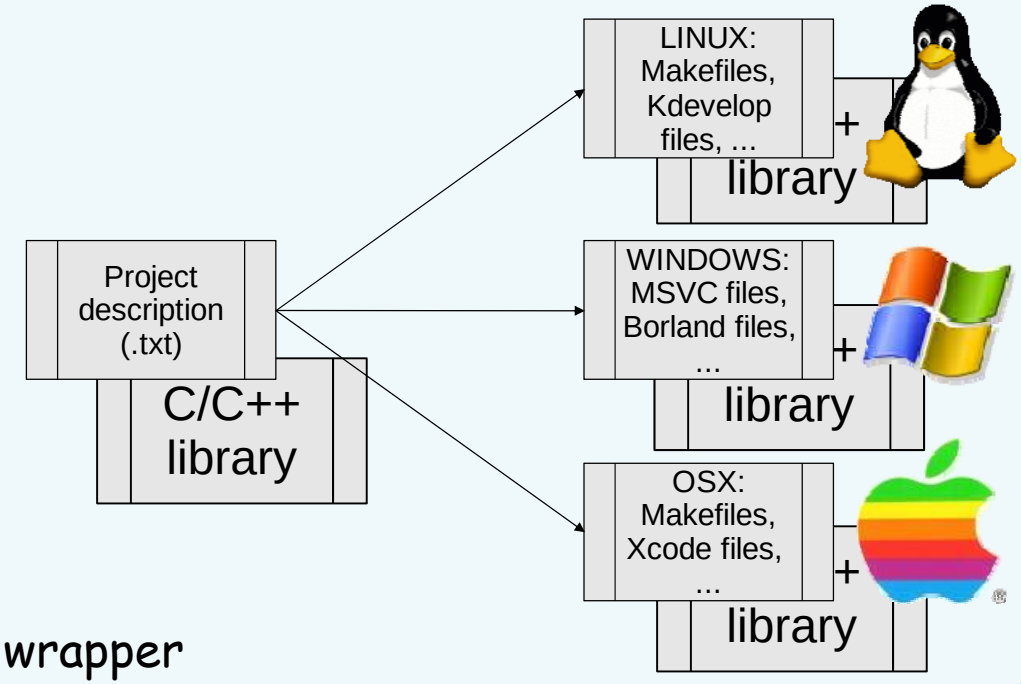


- YARP is an open-source middleware for humanoid robotics
- History
  - An MIT / Univ. of Genoa collaboration
  - Born on Kismet, grew on COG
  - With a major overhaul, now used by RobotCub consortium
  - Exists as an independent open source project
  - C++ source code



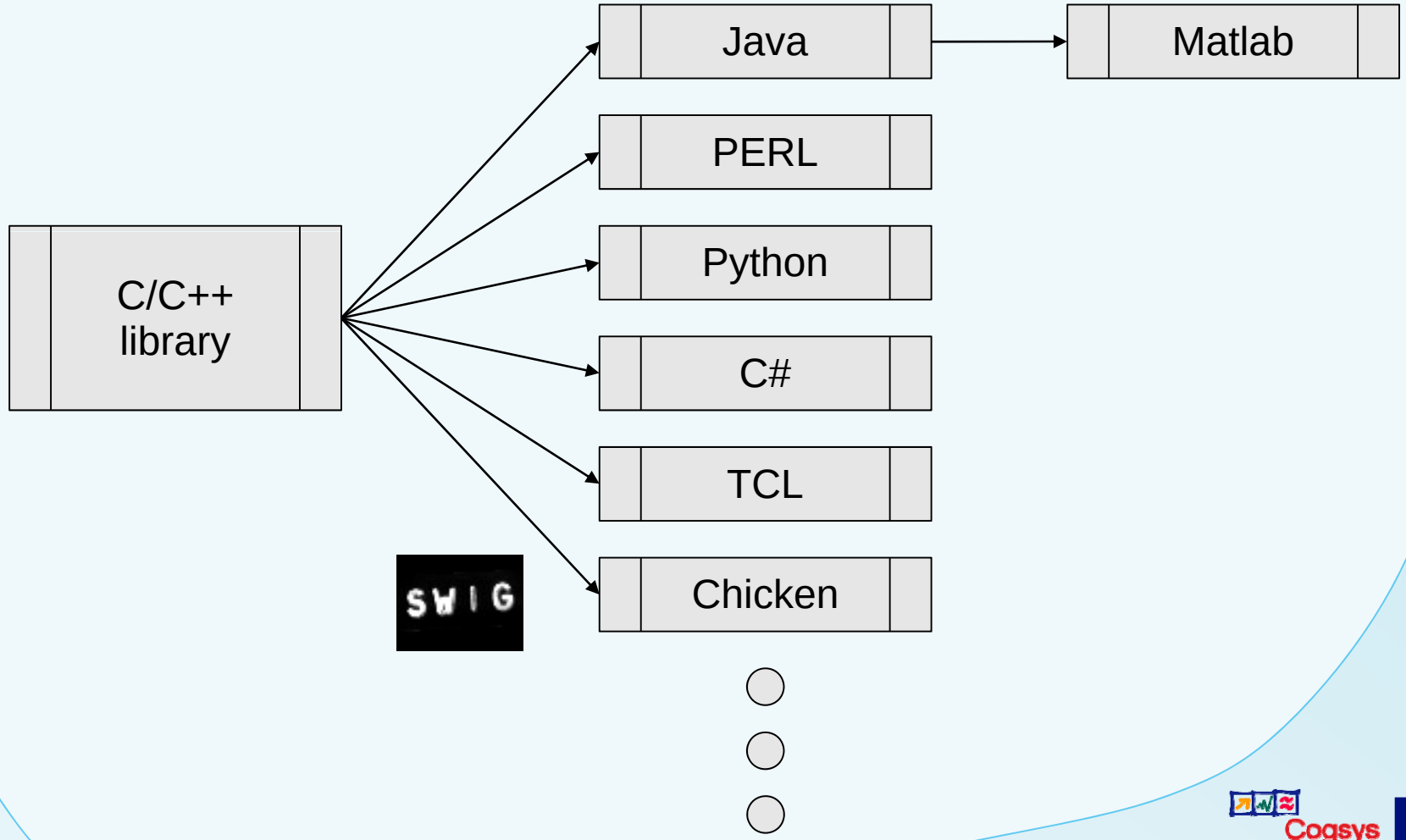
IDE/OS Portability

CMake
Cross-platform Make



OS portability
C++ OS functionality wrapper
e.g. threads, semaphores, sockets

Language Portability (SWIG)



What is YARP for?

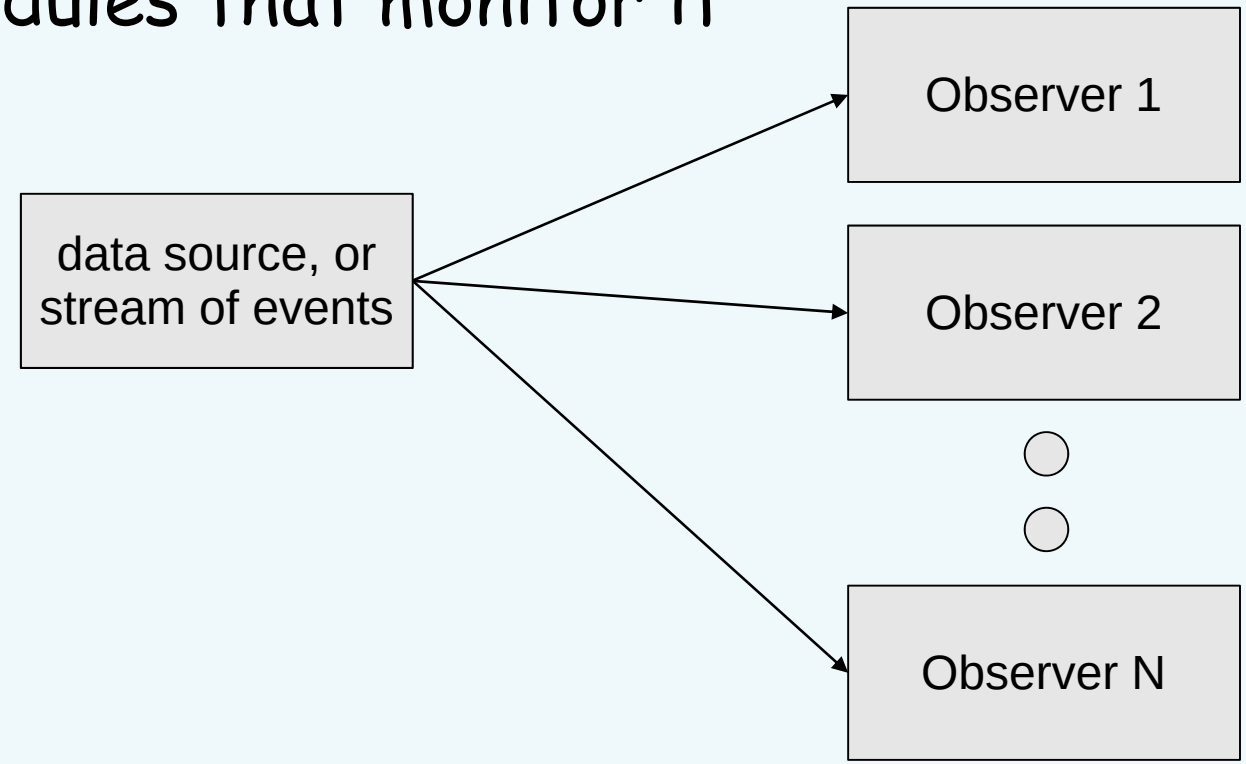
- Factor out **details of data flow between programs** from program source code
  - Data flow is very specific to robot platform, experimental setup, network layout, communication protocol, etc.
  - Useful to keep "algorithm" and "plumbing" separate
- Factor out **details of devices used by programs** from program source code
  - The devices can then be replaced over time by comparable alternatives; code can be used in other systems

What is YARP for?

- Factor out **details of data flow between programs** from program source code
  - Data flow is very specific to robot platform, experimental setup, network layout, communication protocol, etc.
  - Useful to keep "algorithm" and "plumbing" separate
- Factor out **details of devices used by programs** from program source code
  - The devices can then be replaced over time by comparable alternatives; code can be used in other systems

the Observer pattern

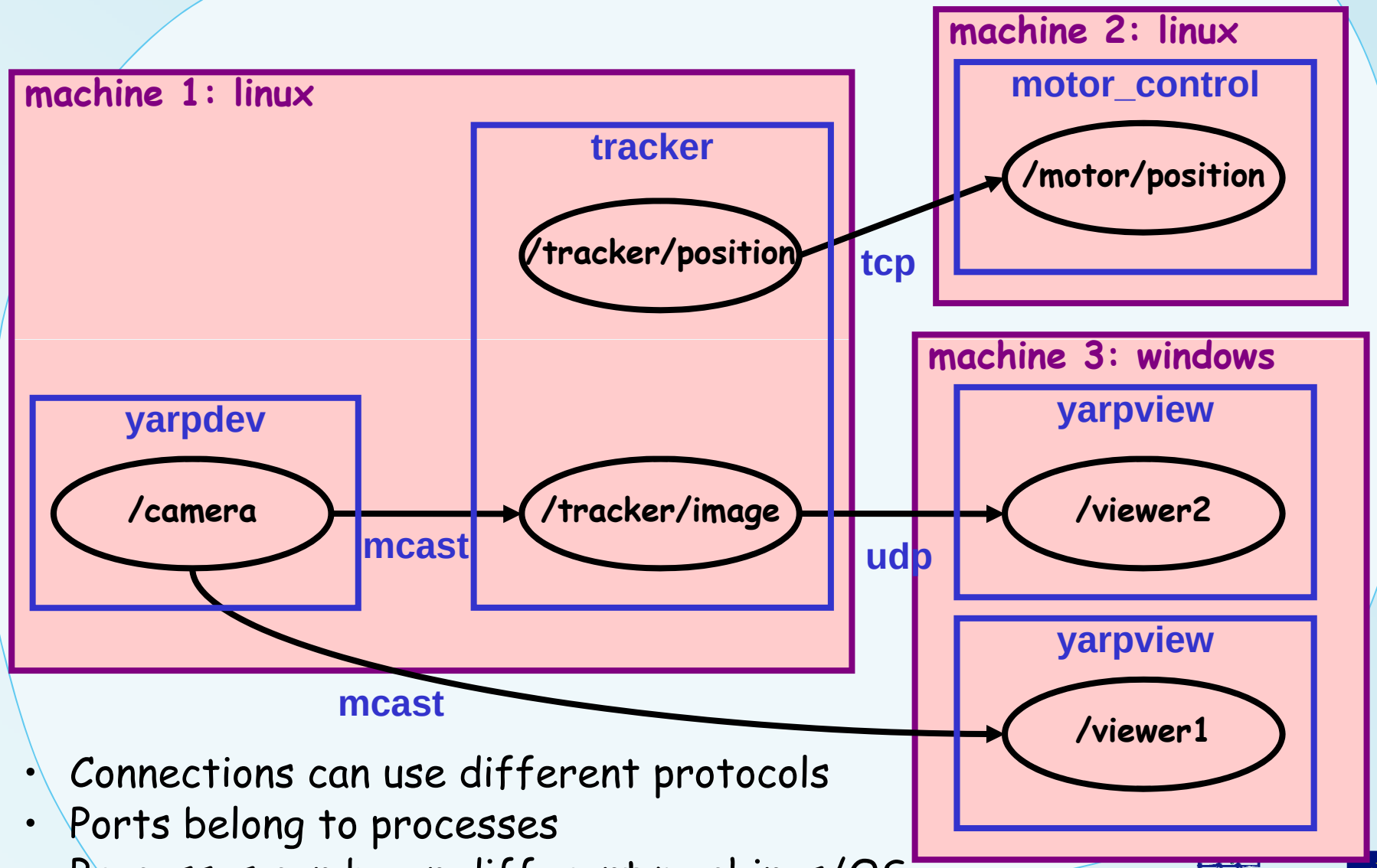
- Data source knows nothing about identity of modules that monitor it



YARP Ports

- We follow the **Observer** design pattern.
- Special "Port" objects deliver data to:
  - Any number of observers (other "Port"s) ...
  - ... in any number of processes ...
  - ... distributed across any number of computers/OSes ...
  - using any of several underlying communication protocols with different technical advantages, streaming or RPC
- This is called the YARP Network

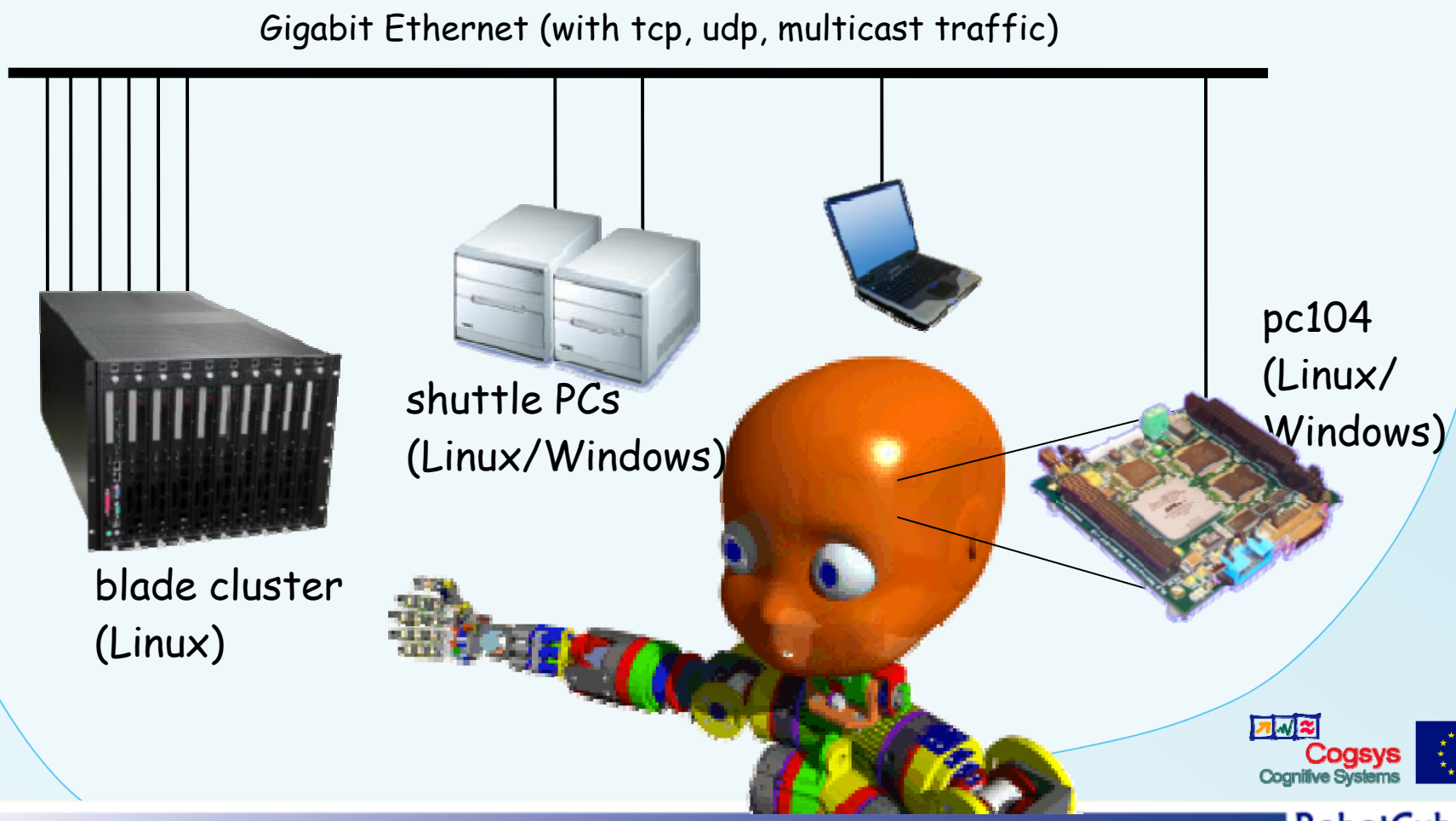
Typical YARP Network



- Connections can use different protocols
- Ports belong to processes
- Processes can be on different machines/OS

Physical Network

Example: RobotCub



Why is all this useful?

- We've separated out most of the plumbing
- We get to change it dynamically (handy)
- More importantly, we have better modularity
  - Programs can be moved around as load and OS/device/library dependencies dictate
  - Fundamental protocol for communication can be changed without affecting programs
  - Better chance that your code can be used by others (even just within your group)