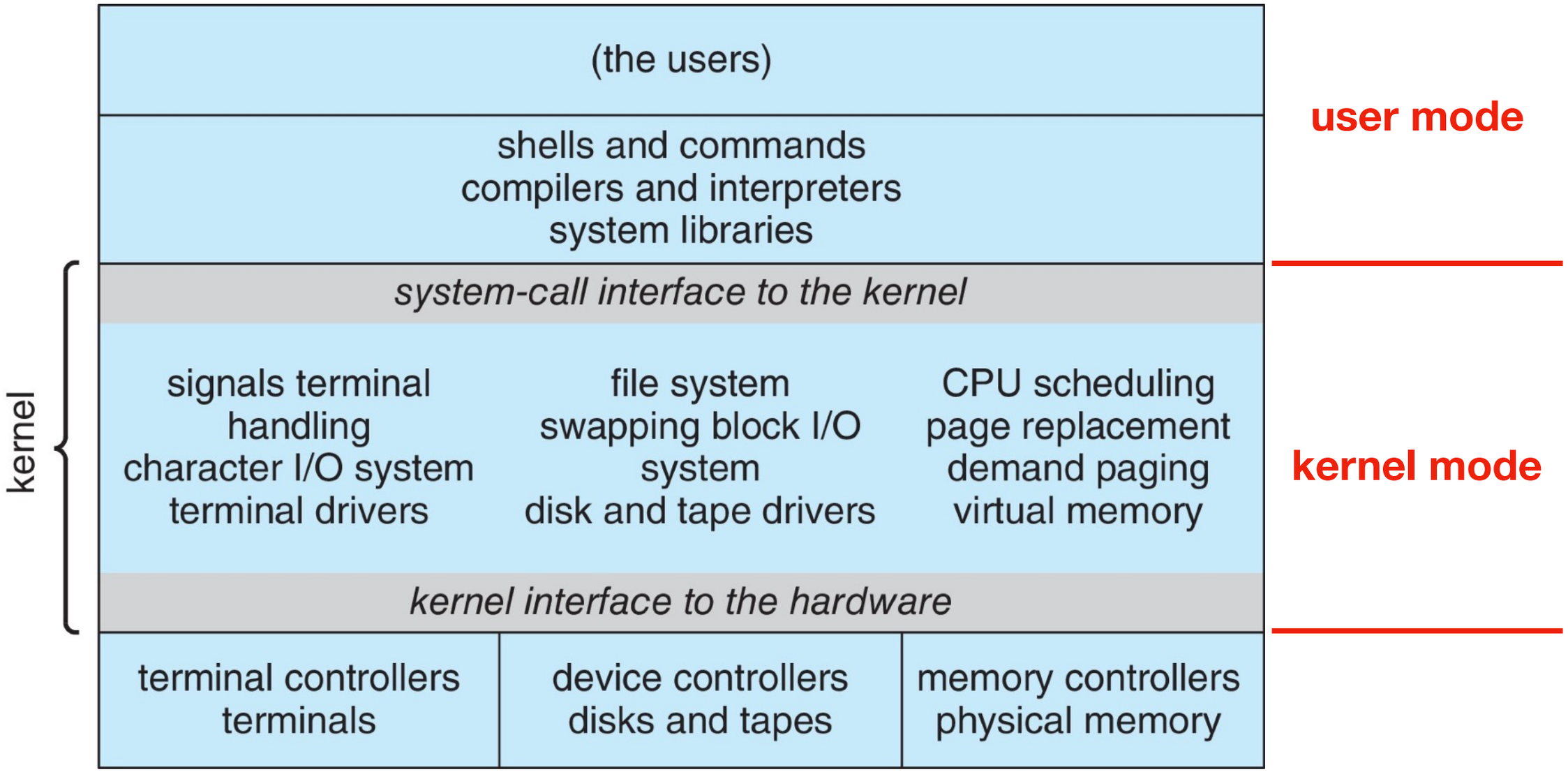
Operating System Concepts

Lecture 4a: Operating System Structure

Omid Ardakanian
oardakan@ualberta.ca
University of Alberta

MWF 12:00-12:50 VVC 2 215

UNIX system has a *monolithic* structure

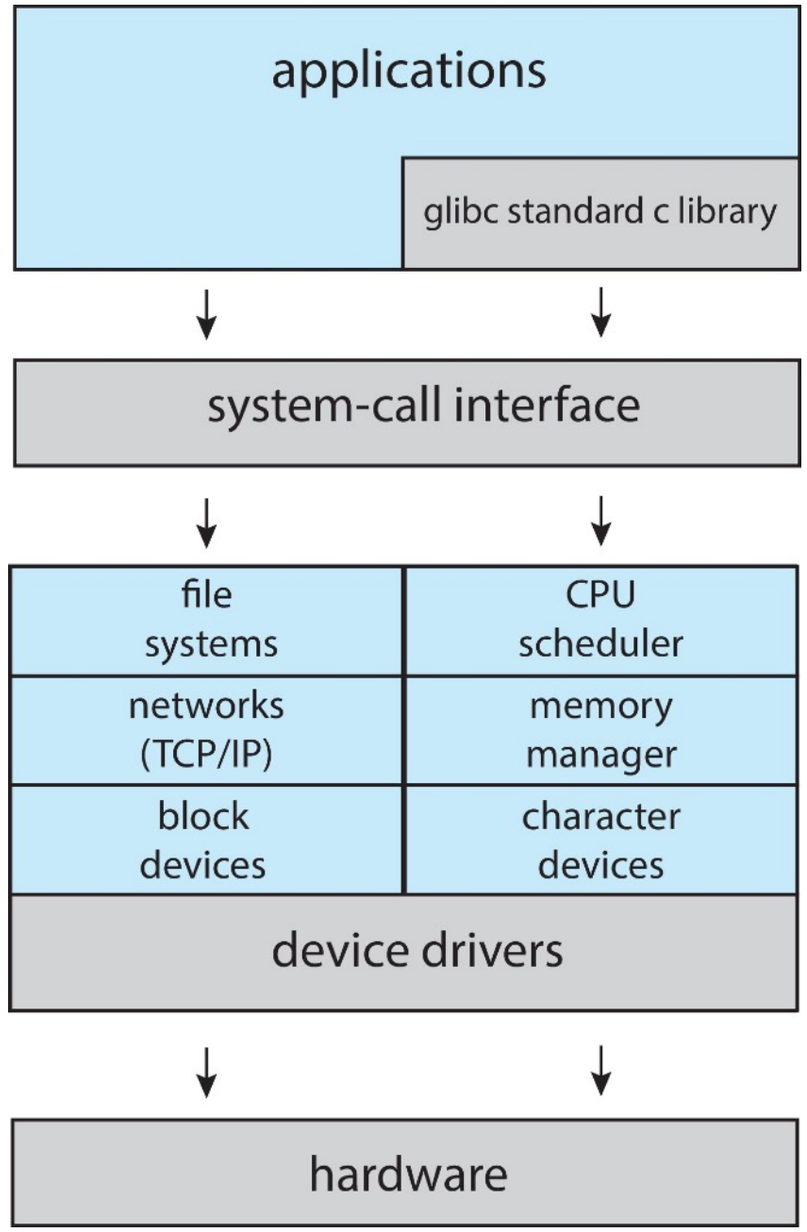


Definition: all functionality of the kernel placed into a single static binary file that runs in a single address space

- faster communication with the kernel
- little overhead in the system call interface

Linux system structure is also monolithic

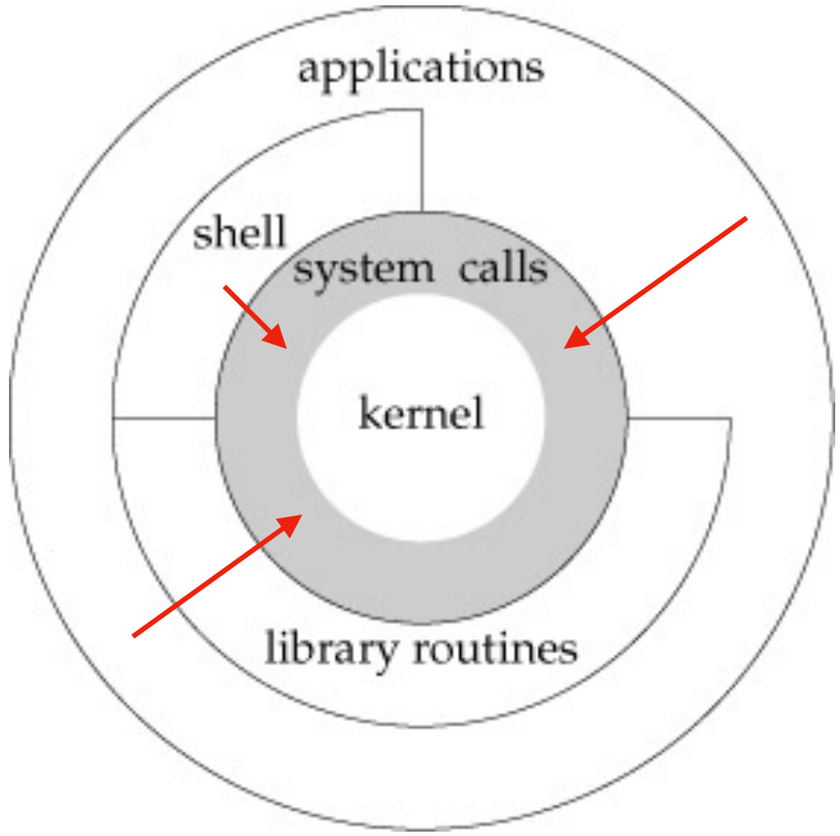
- core (~ 30% of Linux source code) + dynamically loaded kernel modules
  - examples are device drivers, file systems, network protocols
- modules can be loaded at boot time or during run time
  - adding new modules does not require recompiling the kernel
  - each module talks to other modules through known interfaces



Linux system structure is also monolithic

- Applications use the GNU version of the standard C library (**glibc**) which provides the functionality required by POSIX
  - glibc provides a wrapper around system calls
  - glibc is the system-call interface to the kernel
- Linux kernel source: https://www.kernel.org/
  - Directory structure:

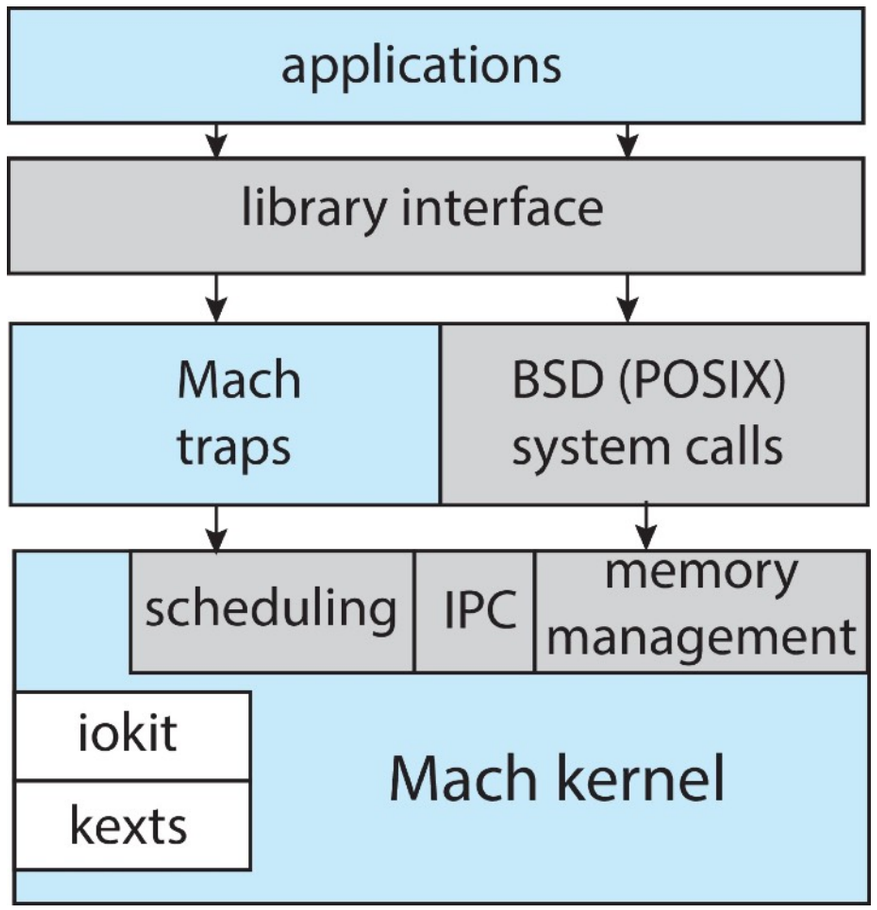
```
include: public headers
kernel: core kernel components (e.g., scheduler)
arch: hardware-dependent code
fs: file systems
mm: memory management
ipc: interprocess communication
drivers: device drivers
usr: user-space code
lib: common libraries
```



Darwin, macOS kernel, has a hybrid structure

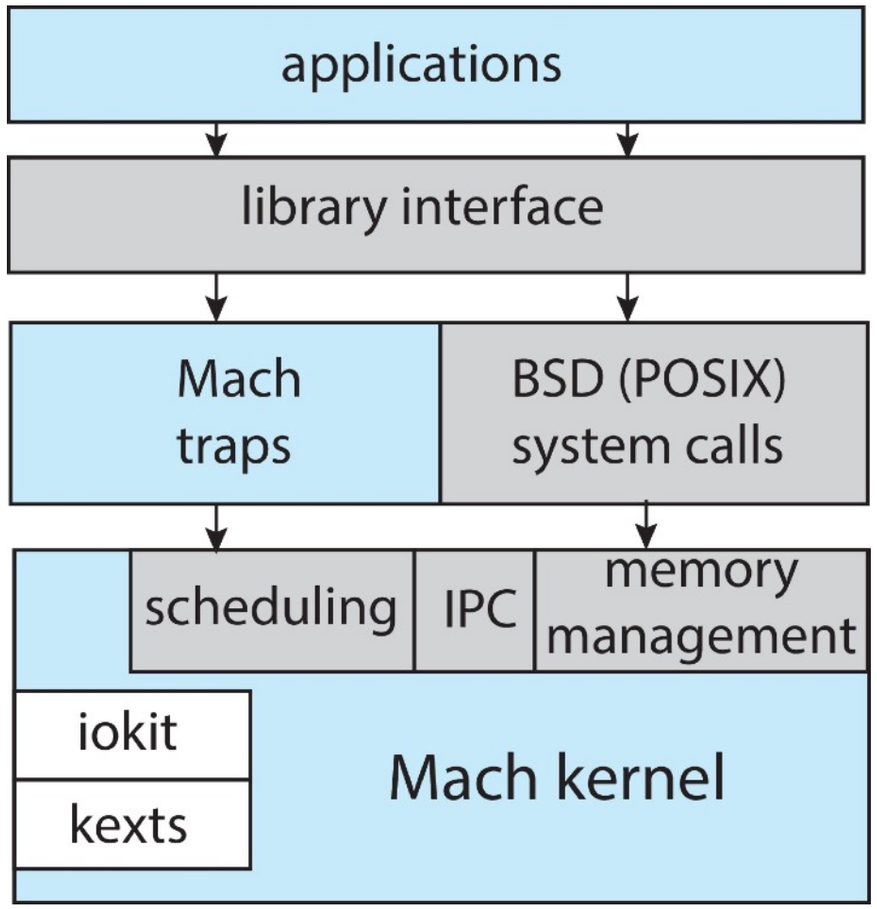
Darwin, macOS kernel, has a hybrid structure

- Darwin is Mach microkernel + BSD (threads, command line interface, networking, file system) + user-level services (GUI)



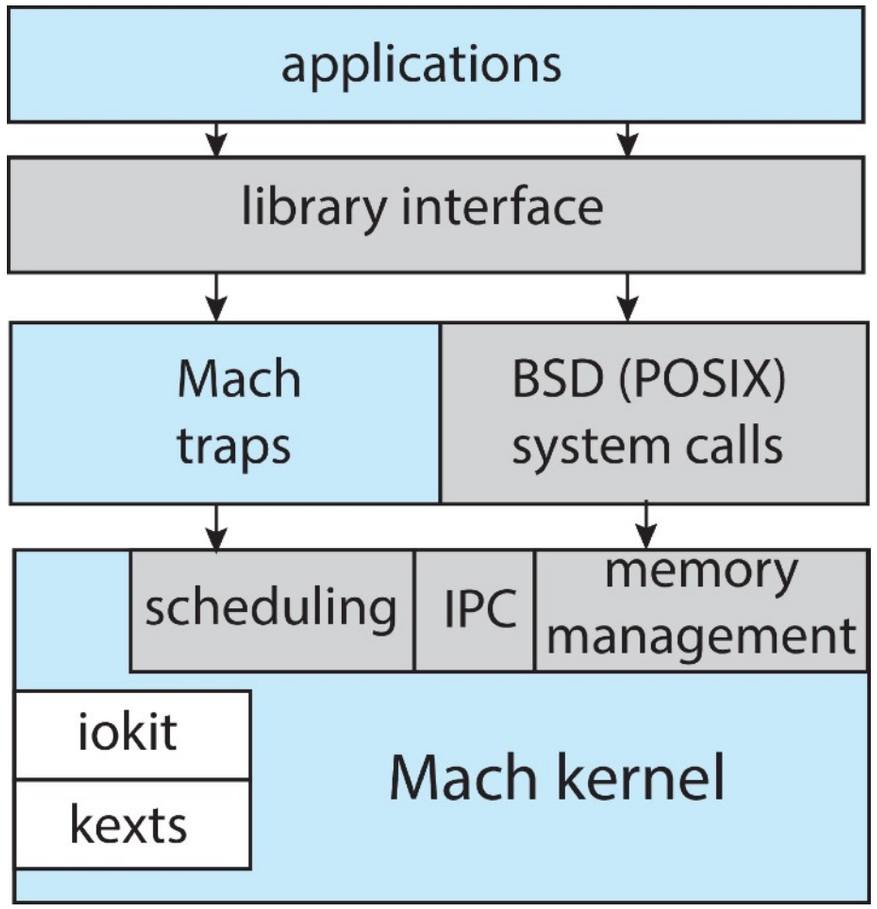
Darwin, macOS kernel, has a hybrid structure

- Darwin is Mach microkernel + BSD (threads, command line interface, networking, file system) + user-level services (GUI)
- Layered structure
  - advantages: modularity, simplicity, portability, ease of design/ debugging
  - disadvantage: communication overhead between layers, extra copying, book-keeping



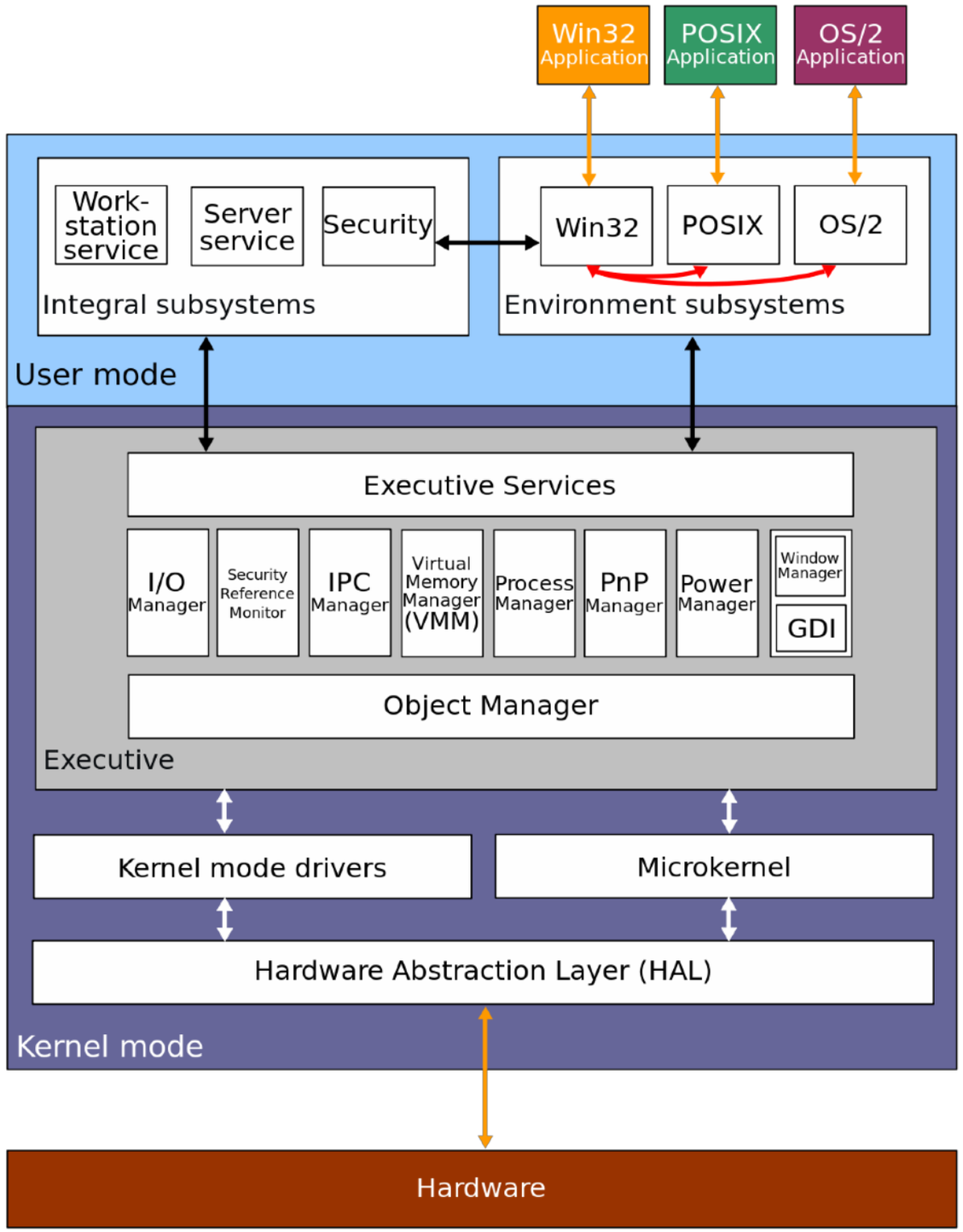
Darwin, macOS kernel, has a hybrid structure

- Darwin is Mach microkernel + BSD (threads, command line interface, networking, file system) + user-level services (GUI)
- Layered structure
  - advantages: modularity, simplicity, portability, ease of design/ debugging
  - disadvantage: communication overhead between layers, extra copying, book-keeping
- Microkernel structure
  - a small kernel providing
    - interprocess communication (message passing usually through **ports**)
    - basic functionality (e.g., scheduling and virtual memory management)
  - other OS functionality (device drivers, file system, networking, user interface, etc.) implemented as user-space processes



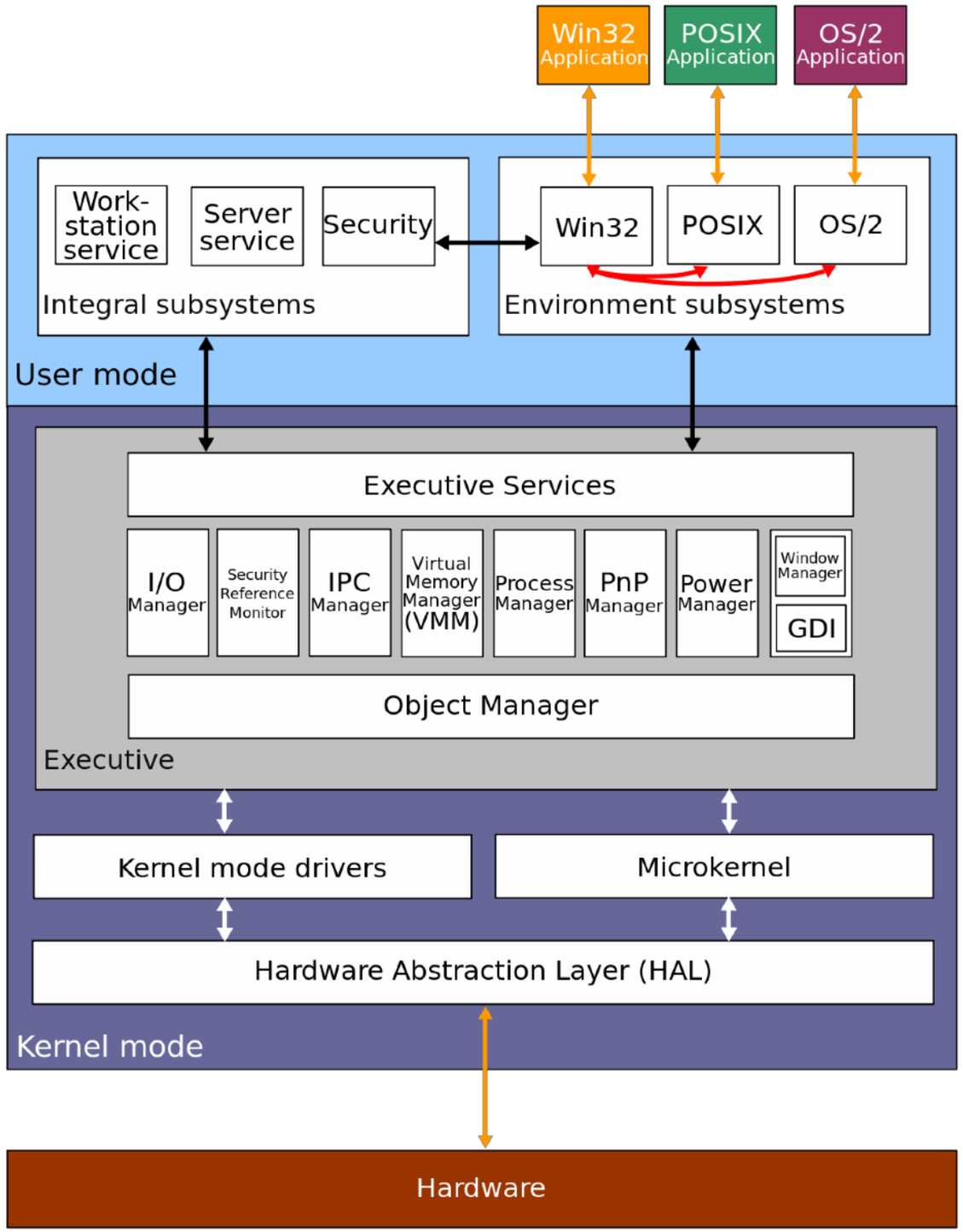
Windows NT

- Layered architecture with several modules
- the Windows executive provides core OS services (Ntoskrnl.exe)
- the kernel itself resides between HAL and the executive layer
  - responsible for thread scheduling and interrupt handling



Windows NT

- Layered architecture with several modules
- the Windows executive provides core OS services (Ntoskrnl.exe)
- the kernel itself resides between HAL and the executive layer
  - responsible for thread scheduling and interrupt handling
- Windows native APIs are undocumented
  - POSIX, OS/2 and Win32 APIs are documented and can be used by applications



Windows NT

- Layered architecture with several modules
- the Windows executive provides core OS services (Ntoskrnl.exe)
- the kernel itself resides between HAL and the executive layer
  - responsible for thread scheduling and interrupt handling
- Windows native APIs are undocumented
  - POSIX, OS/2 and Win32 APIs are documented and can be used by applications
- the hardware abstraction layer (HAL.dll) provides portability across a variety of hardware platforms
  - device drivers call functions in the HAL to interface with the hardware