Abstract Machines in OS

Peter J. Denning

Rules

Levels like stacked blocks

Each manages a class of OS objects

Each has user interface (API)

Our entire kernel stack is 9 blocks

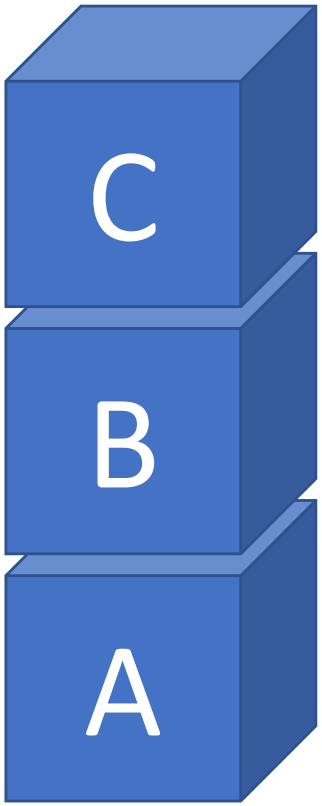
Kernel API = union of 9 block APIs

Protected calls (forced entry points to API functions)

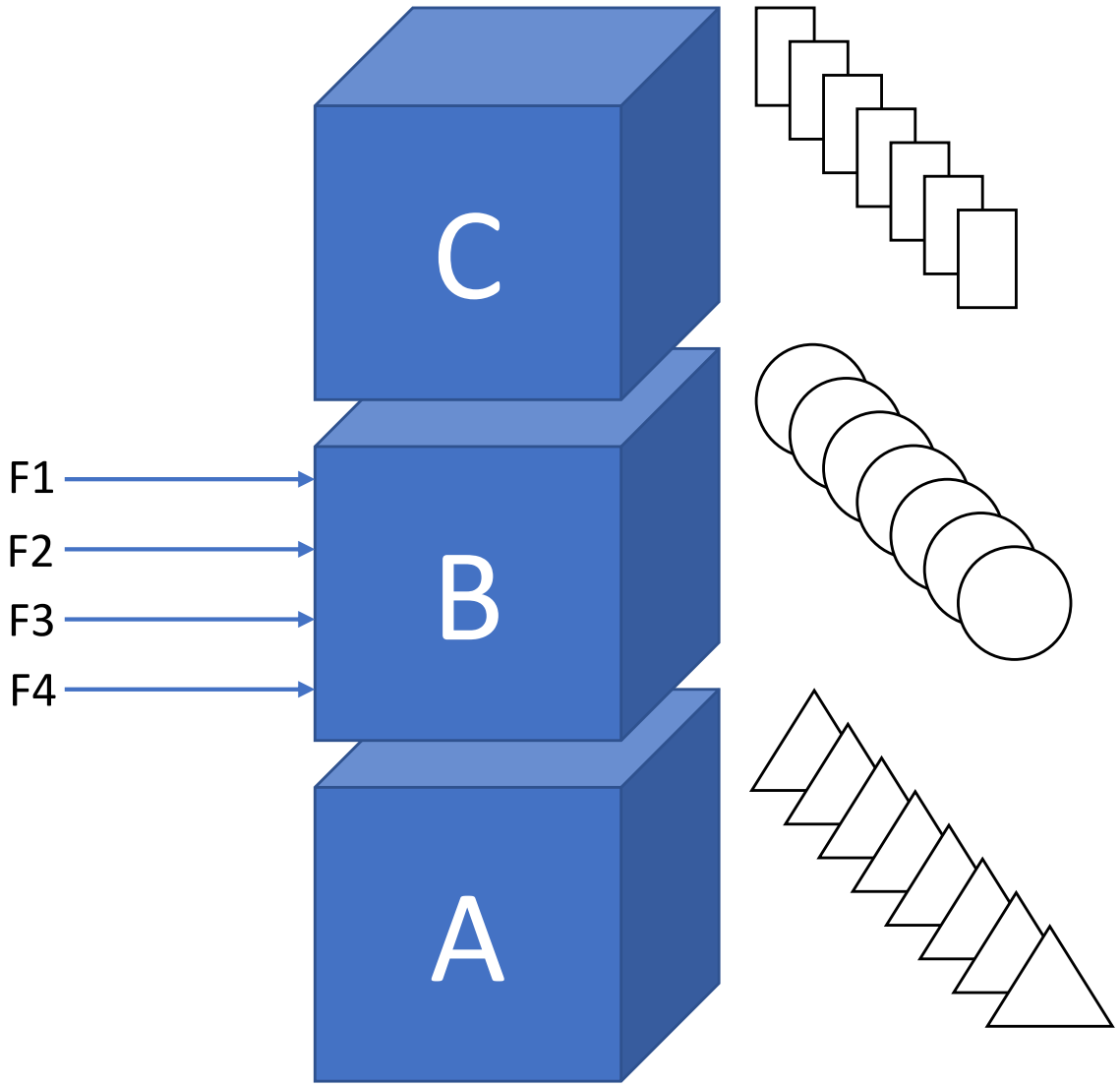
Each can use components offered by blocks below in the stack but no knowledge of anything above

Downward calls, upward returns

Each block has private kernel memory inaccessible to users

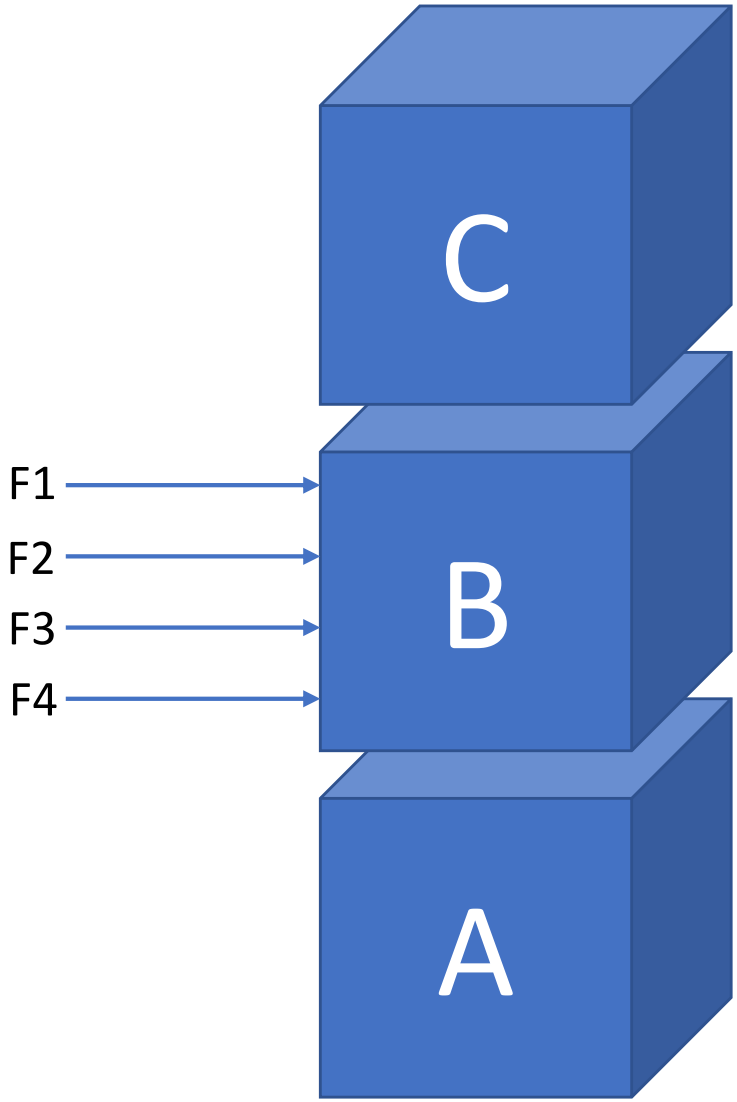


API is set of function names that can be called to do operations on objects

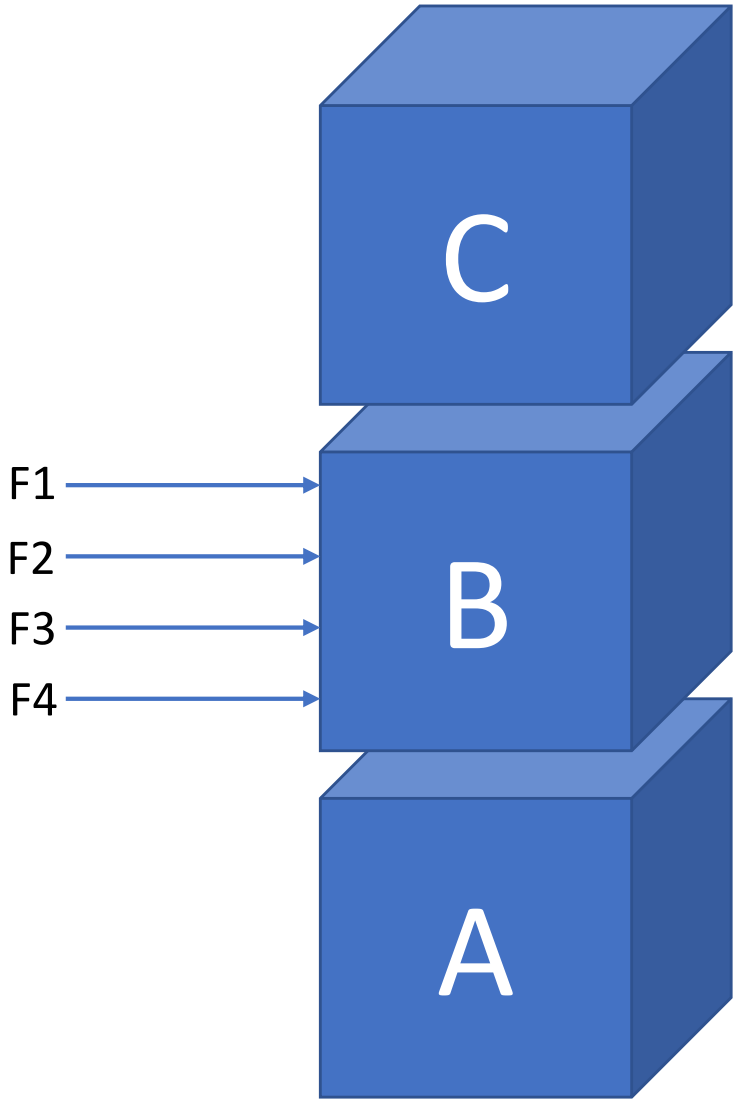


each level manages objects of the same kind

User says “F1(c)” to tell the kernel to perform operation F1 on object c



All the functions are called by "protected entry". They always begin executing from their first instructions



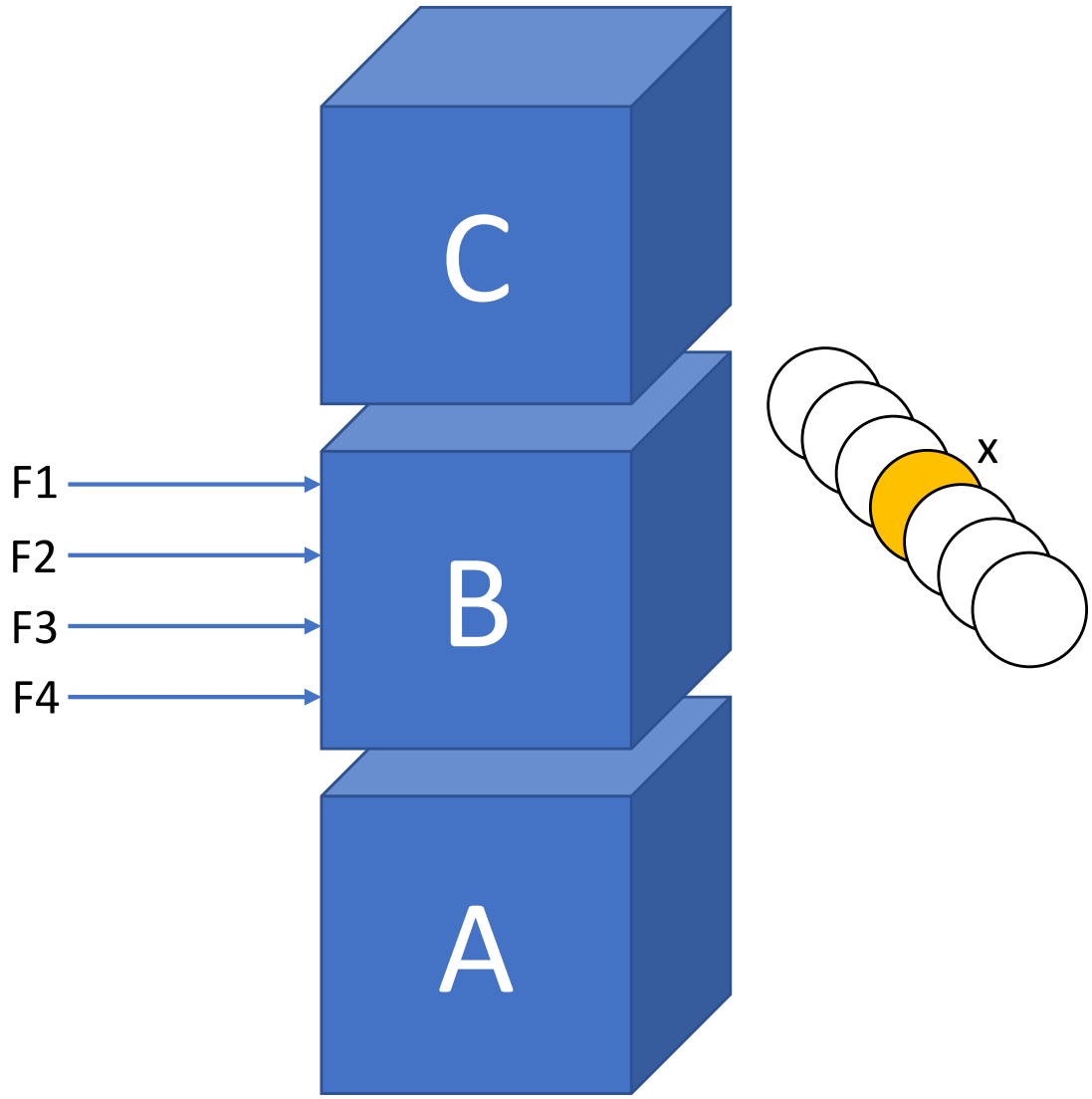
In call F1(c), c is pointer to the target object.

c contains a handle x for the object, a unique for all time name.

c contains a tag “B” indicating that only level B is authorized to do operations on c

c contains access code A set by creator indicating which functions of the API are allowed. A is 4 bits, each enabling one of the 4 functions.

c = (B,A,x) is a capability encapsulating the above. It must be unforgeable.

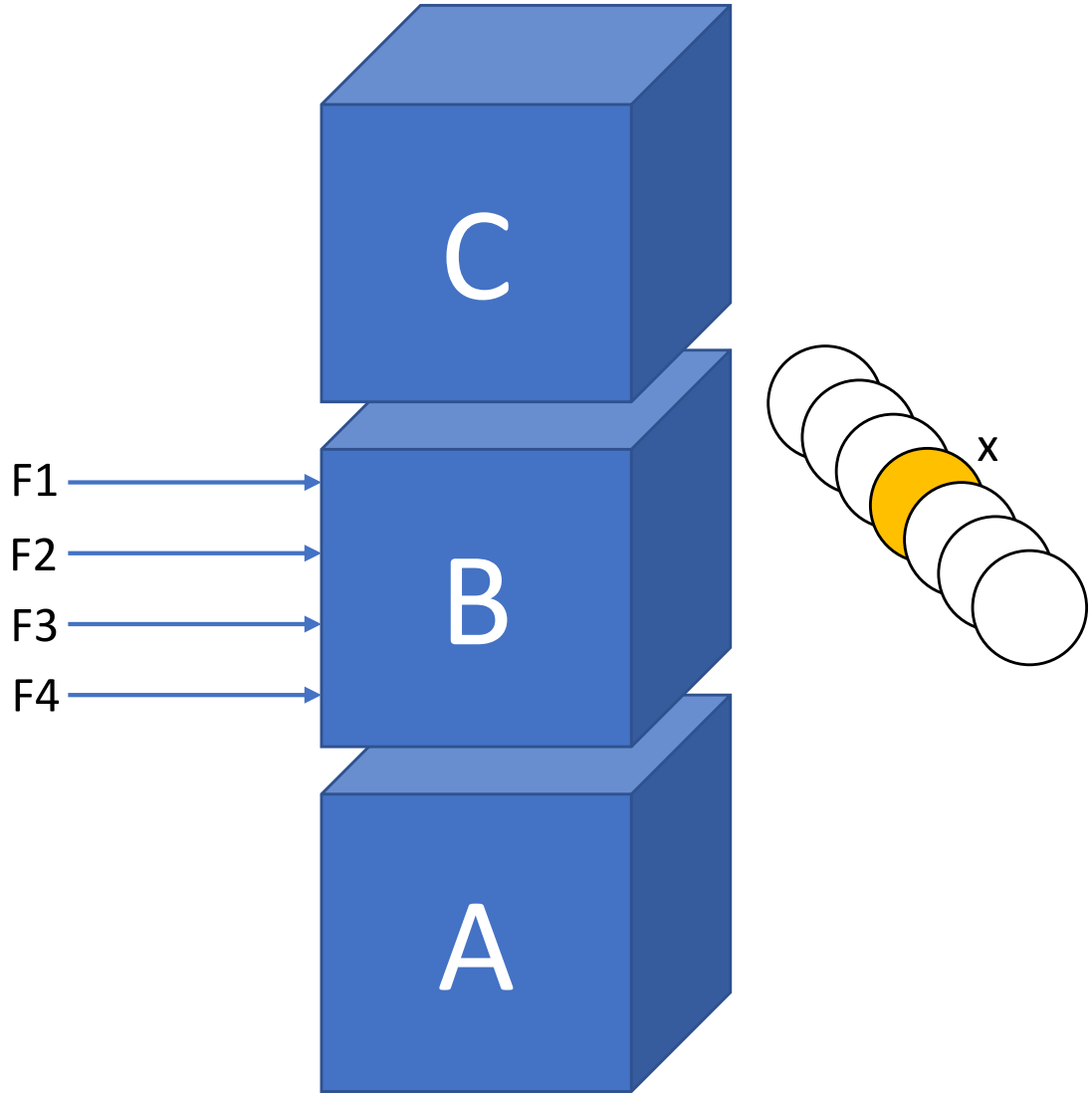


For the call F1(c) to succeed:

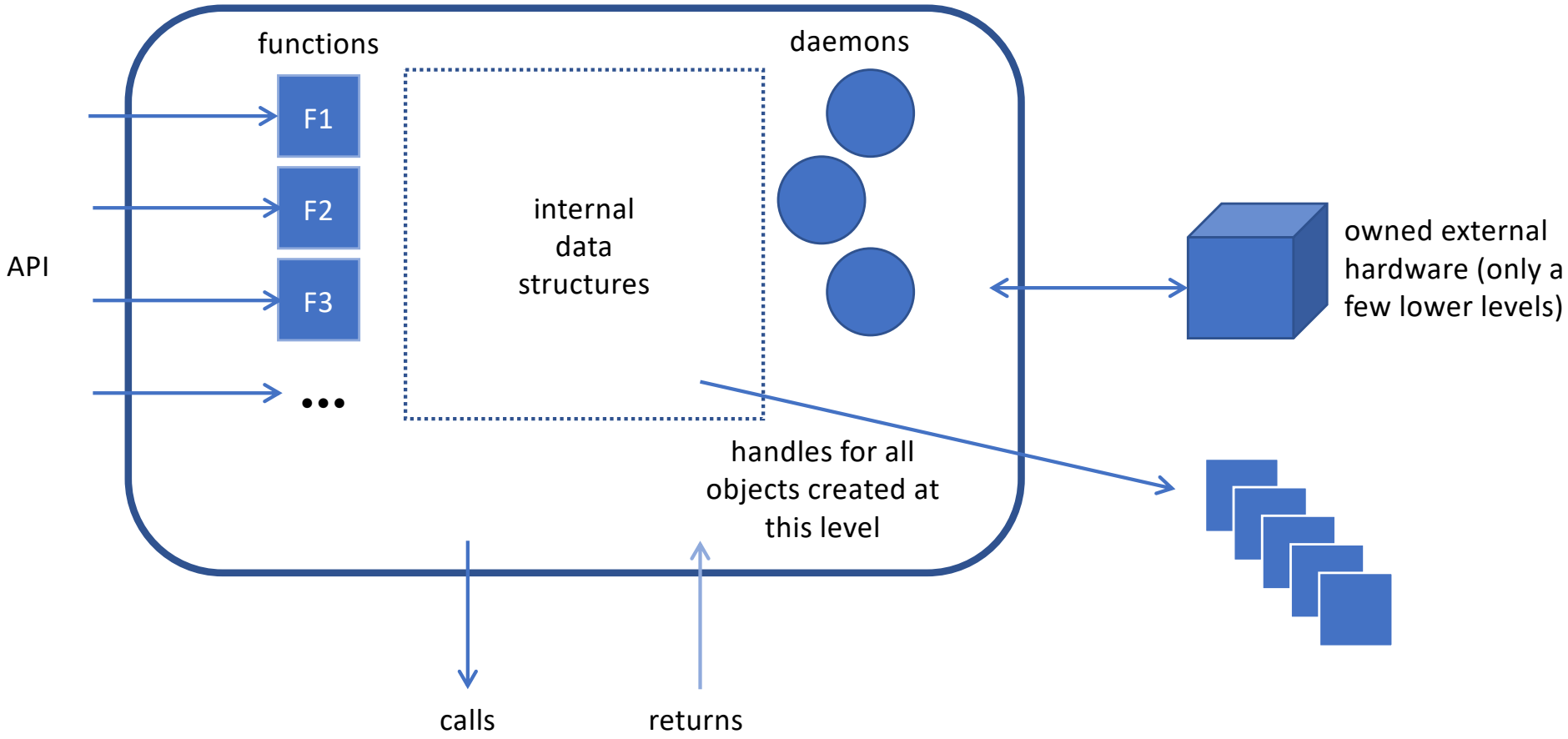
B verifies tag is “B”

B verifies that A allows F1

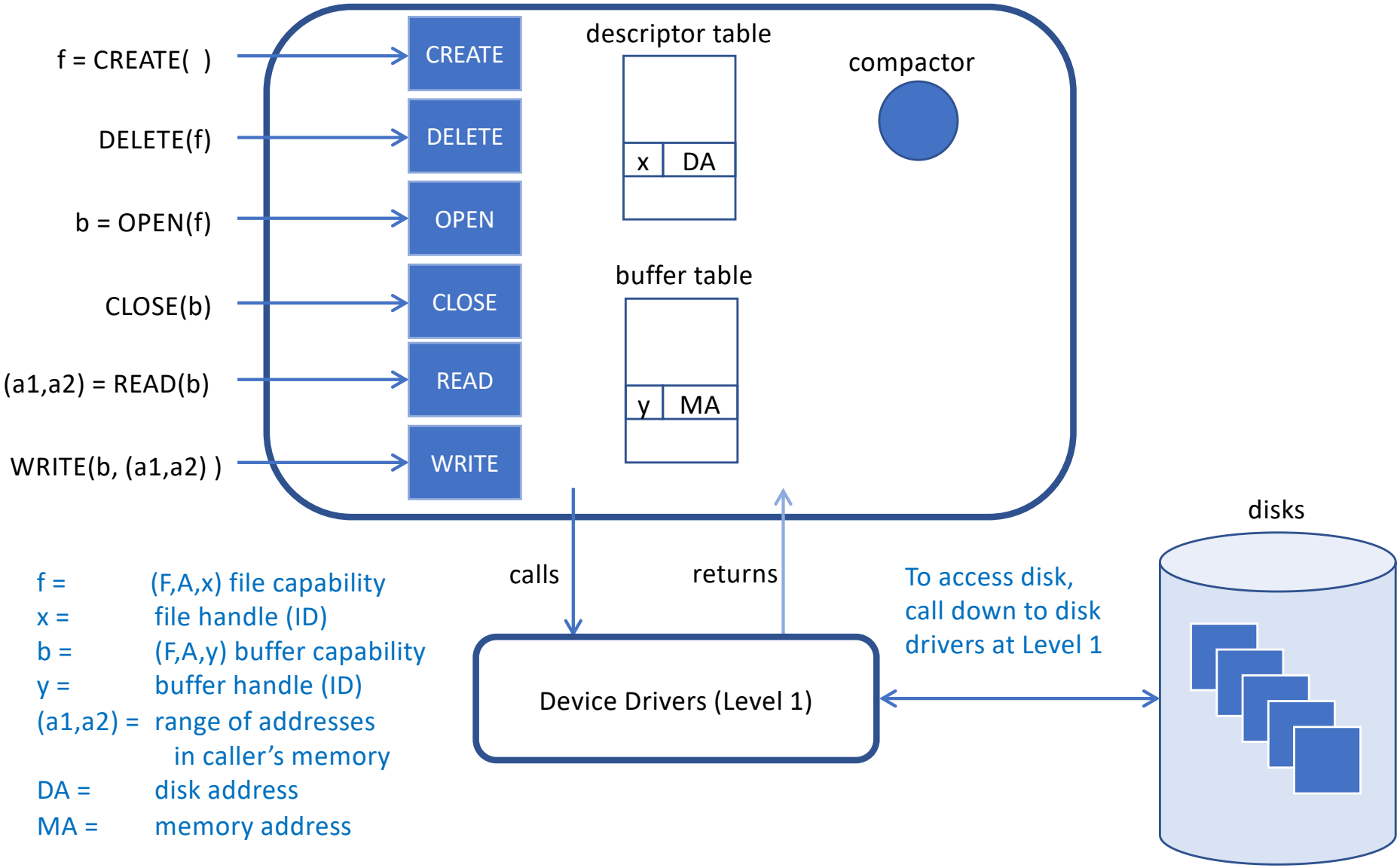
Only then does B map handle x to the object and perform F1 on it.

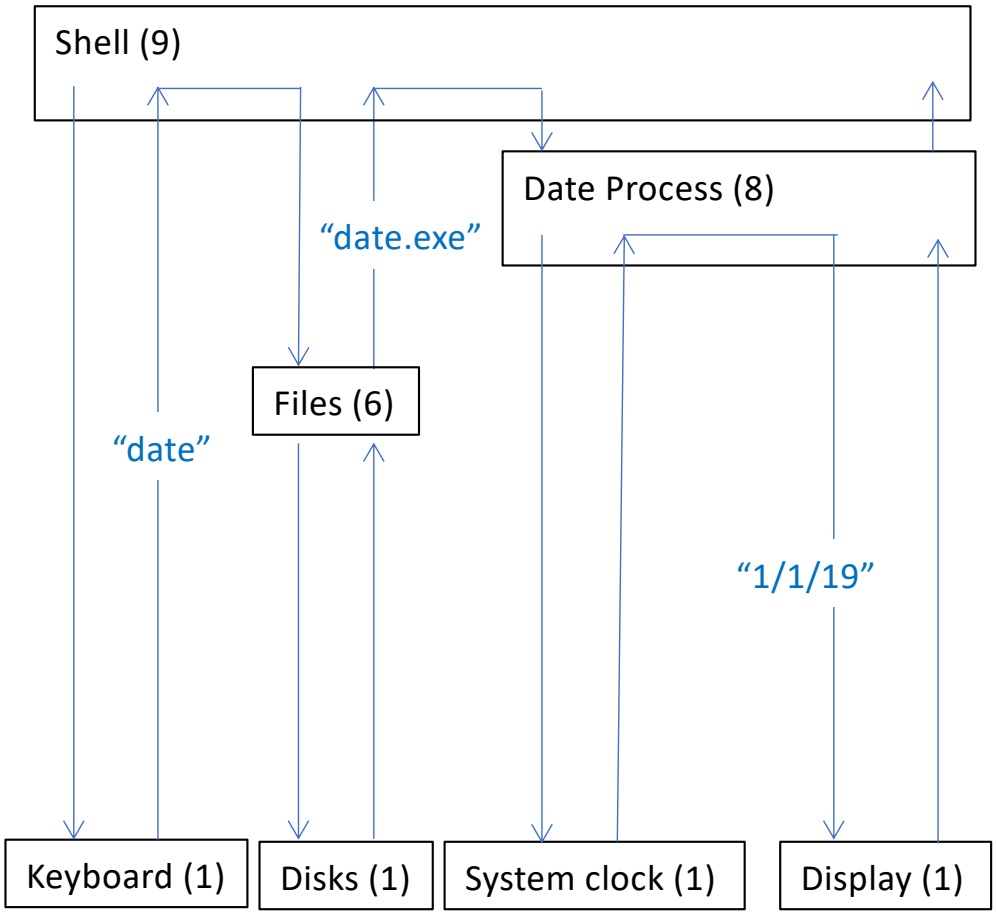


COMPONENTS OF A LEVEL



EXAMPLE FOR FILE MANAGER





Downward calls and upward returns involved in executing the "date" command. Numbers indicate level called.

Shell listens to keyboard for string "date" naming a command, passes it to process manager which gets the executable file "date.exe" and builds a process (virtual machine), which reads the system clock, formats the reading, passes it to the display. When all is done, the lower level managers return to the shell, which starts listening for the next command.