Using a Linux Security Module for contest security

Bruce Merry

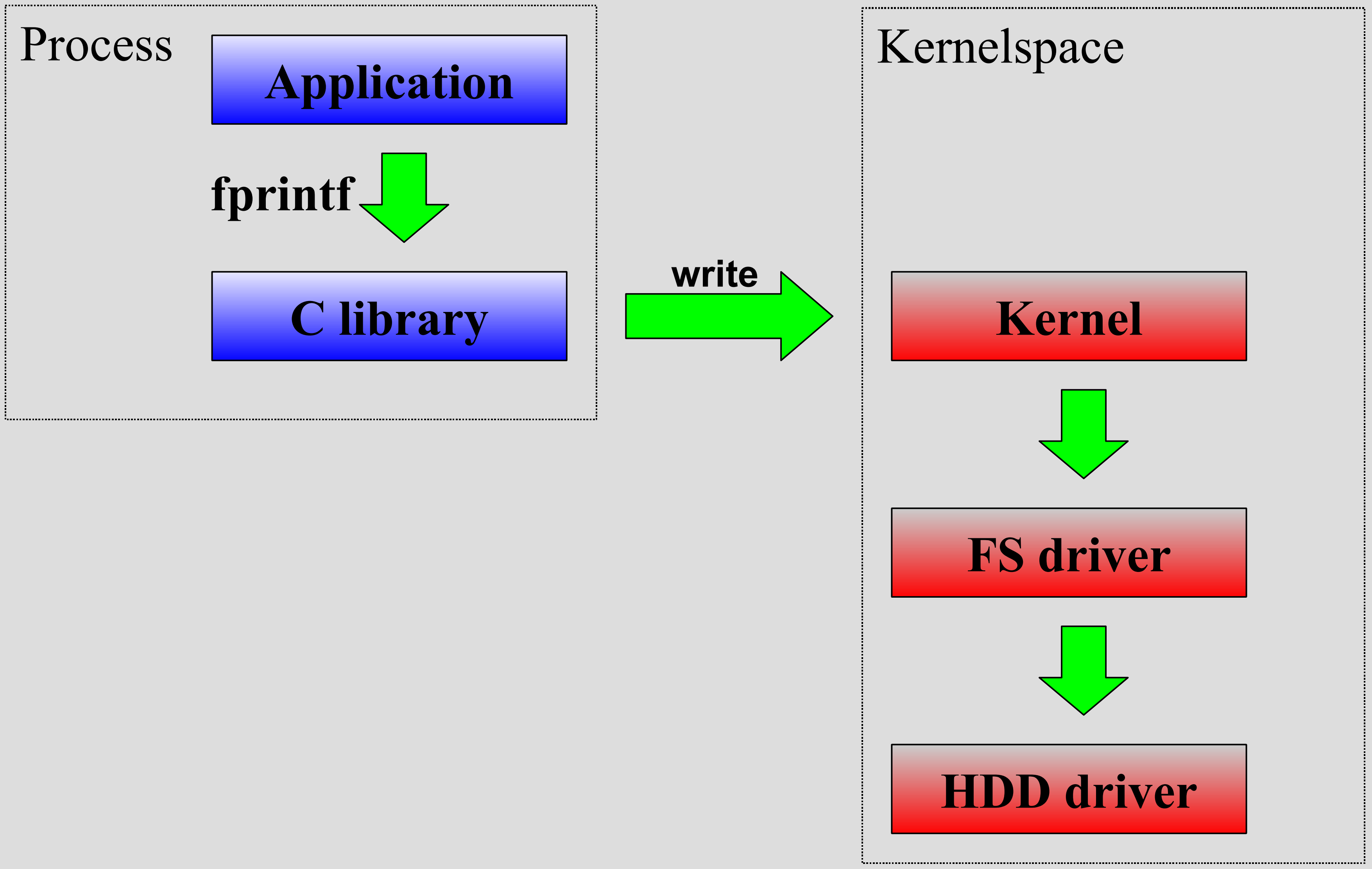
Contents

The goals

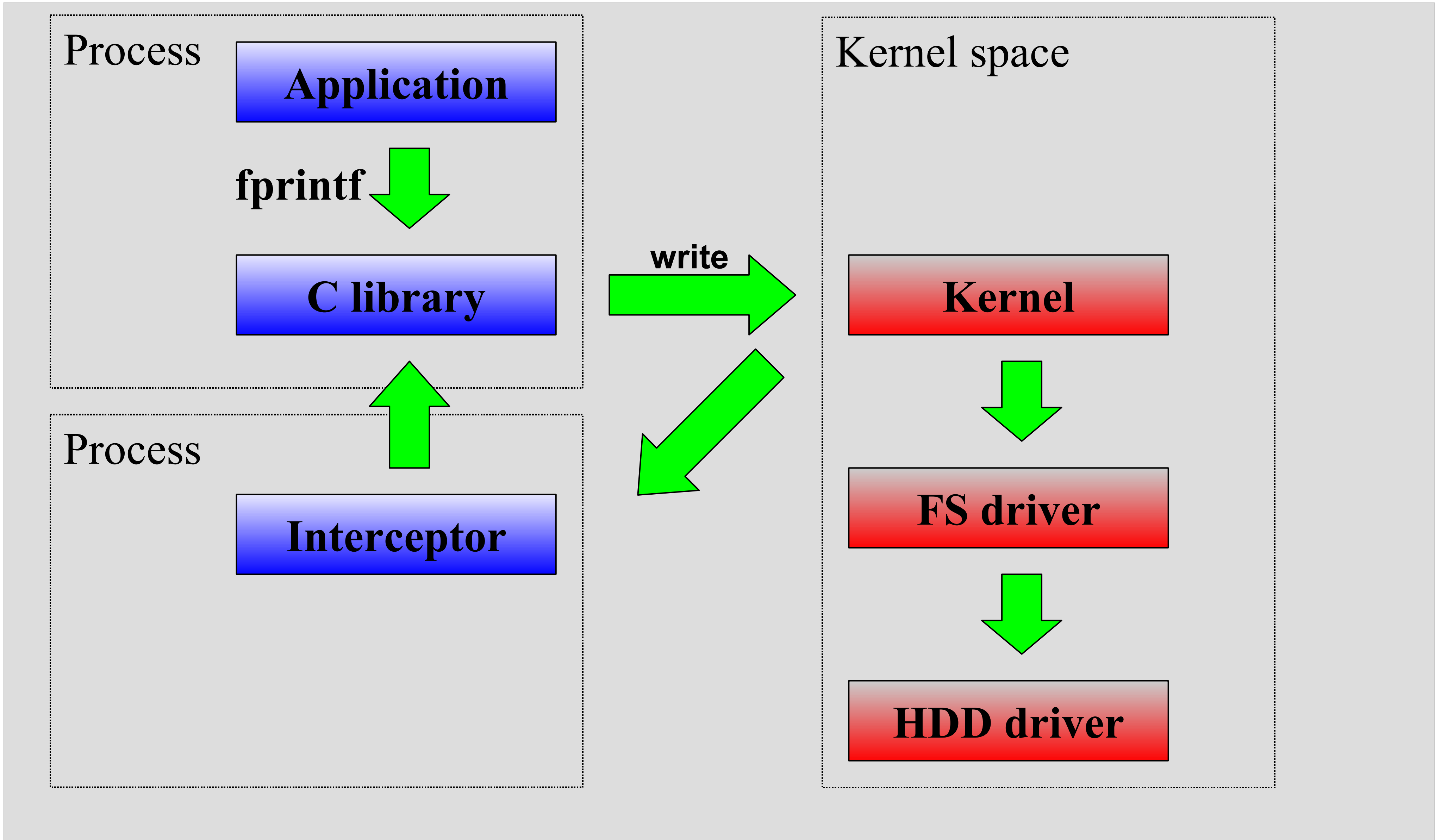
- Resource limits
- No networking
- No IPC
- No access to evaluation system
- Single process
- Single thread

- Accurate constraints
- High throughput
- Minimum overhead
- Transparent

Device access in Linux



System call wrappers



System call wrappers

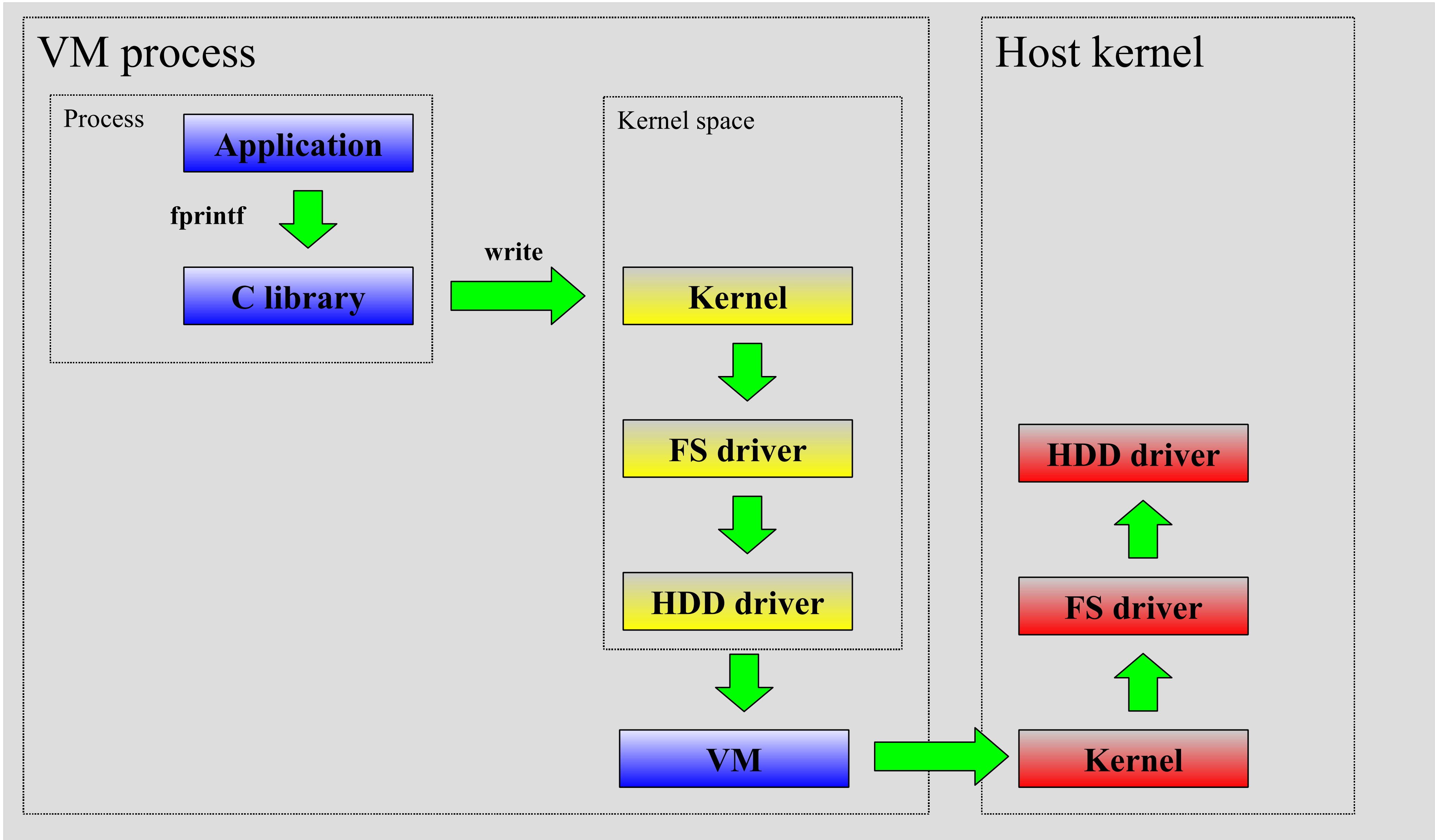
Pros

- Configurable, off-the-shelf wrappers available
- Minimal startup overhead

Cons

- Context switch per system call
- Huge number of system calls
- Poor security track record

Virtualisation



Virtualisation

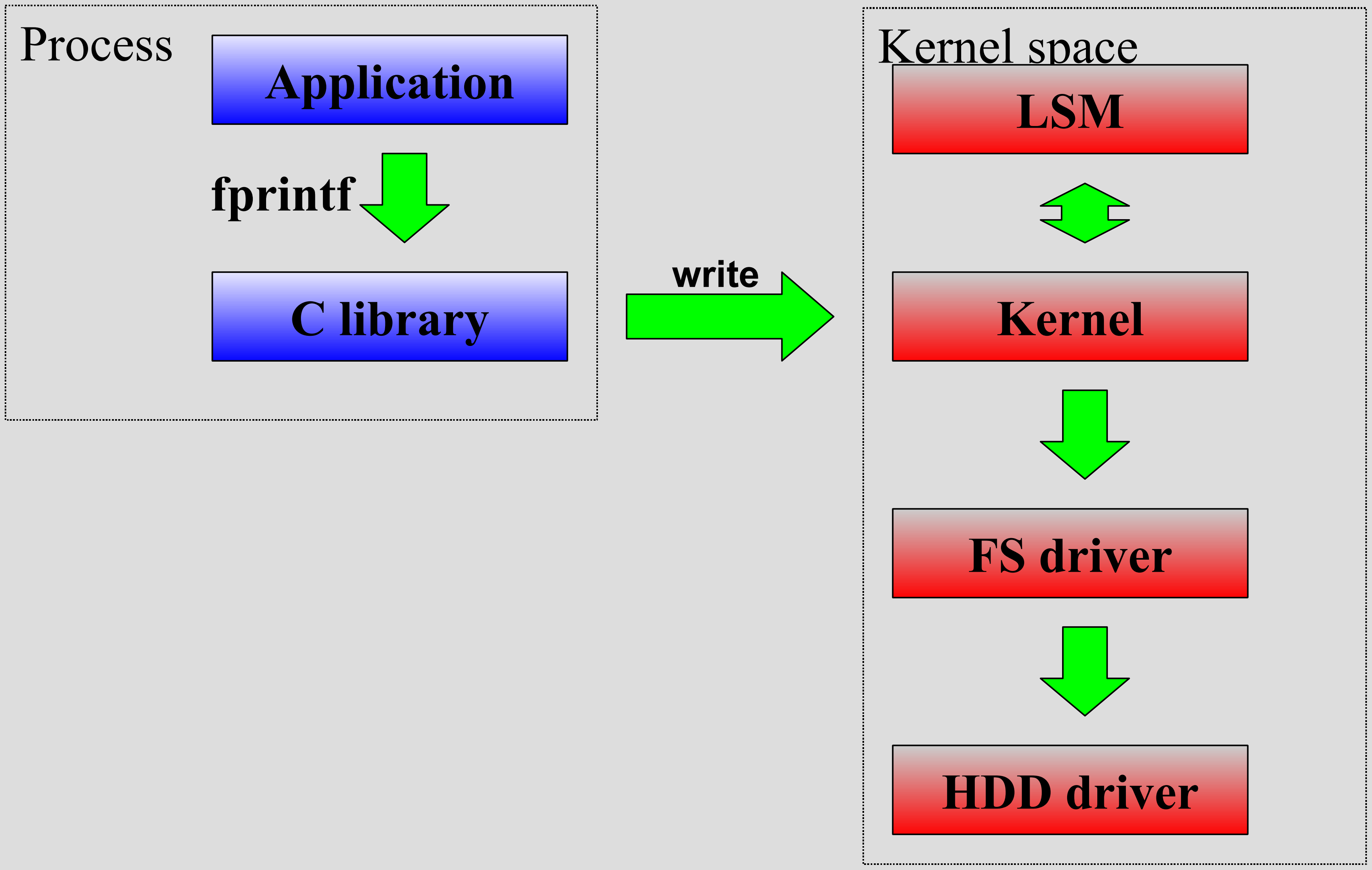
Pros

- Guest OS can be totally isolated
- Can start with a totally fresh OS for each run

Cons

- Performance impact
- Startup time
- Does not prevent multi-threading, external processes etc.

Linux Security Module



Linux Security Module

Pros

- Policy per operation, not per syscall
- No extra context switches
- Access to kernel internals
- Fewer races

Cons

- Kernel programming is difficult
- Interface changes frequently
- Outdated docs

LSM implementation: sandtray

- Launcher program requests restrictions:
  - Calls `setrlimit` to set CPU, memory etc limits
  - Writes to `/proc/self/attr/exec` to set further limits:
    - `version 1.0` (sets default restrictions)
    - `allow write problem.out`
- Launcher then calls `exec`
  - This triggers sandtray for this process
- Caller asks for exact CPU time on return
  - `setrlimit` only has 1 second resolution

Filesystem access

- glibc accesses huge numbers of files
  - A whitelist is difficult to maintain
  - Path-based checks tricky due to links
- Instead, read access left open
  - Contest internals owned by a different user
- Write access is tightly controlled
  - Only the output file may be written
  - Together with `setrlimit`, limits total disk space
  - No symlinks, chmod, chown, etc.

Covert channels

- Sandtray cuts off
  - Networking, SystemV IPC, kill etc.
  - Writing files into `/tmp` or similar
- Are still channels through `/proc` and others
  - Now prevented by serialising execution
- Cache timing theoretically possible
  - Probably harder than solving the original problem

Java (Sun VM)

- Consumes huge amounts of virtual memory
- Does lots of suspicious-looking things
- Has its own security manager
  - `permission java.io.FilePermission …`
- Has command-line option for max heap size:
  - `-Xmx 64m`
- We use these instead of sandtray

Conclusions

- LSM
  - Good abstraction of operations
  - Low overhead
  - Interface is a moving target
- Sun Java VM
  - VM does not play nicely with LSM
  - Internal security tools are good enough

Questions?