AOSV: Final Project Assignment

Alessandro Pellegrini

A.Y. 2017/2018

User-Level Threads

- User-Level Threads (ULTs) are execution contexts which live within a single thread
- A ULT is non-preëmptive
  - Switching from one ULT to the other requires the currently-running one to yield the CPU
  - Often also called *co-routines*
- This is not traditionally implemented at kernel-level
- Userspace libraries implement context switches based on `setjmp/longjmp`

Fibers

- Fibers are the Windows kernel-level implementation of User-Level Threads
  - see on MSDN the description
  - see an open source implementation (thanks to ReactOS)
- `ConvertThreadToFiber()`: creates a Fiber in the current thread. From now on, other Fibers can be created
- `CreateFiber()`: creates a new Fiber context, assigns a separate stack, sets up the execution entry point (associated to a function passed as argument to the function)

Fibers

- `SwitchToFiber()`: switches the execution context (in the caller thread) to a different Fiber (it can fail if switching to a Fiber which is already active)
- Fiber-Local Storage (FLS):
  - TLS is shared across fibers (this is related to the thread where the fiber is running)
  - `FlsAlloc()`: Allocates a FLS index
  - `FlsFree()`: Frees a FLS index
  - `FlsGetValue()`: Gets the value associated with a FLS index (a long)
  - `FlsSetValue()`: Sets a value associated with a FLS index (a long)

Project General Rules

- The final project entails implementing fiber support in the Linux kernel
- The choice of the kernel version to use is up to the student (if you can support multiple versions, it's better!)
- How to implement the subsystem is up to the student: a module, a set of new syscalls, both...
- Projects should be developed alone or in groups of 2 students

Project Assignment

- Implement facilities logically related to:
  - `ConvertThreadToFiber()`
  - `CreateFiber()`
  - `SwitchToFiber()`
  - Implement also Fiber-Local Storage
  - Provide a userspace library to access the new services
- This must be correct also on SMP
- Multiple processes can rely on Fibers at the same time
- If relying on a module and a special device is created, it must be correctly exposed in `/dev`

Project Assignment

- In `/proc`, under the pid of the process, a subfolder `fibers/` should be created, with an attribute file for each active Fiber
- The minimal amount of information to show in `/proc` is:
  - whether the Fiber is currently running or not
  - the initial entry point for the created fiber
  - the thread id from which the Fiber was created
  - the number of current activations of the Fiber
  - the number of failed activations
  - total execution time in that Fiber context

What to hand out

- The code
- An essay discussing:
  - Design choices
  - Implementation details
- Some performance assessment
  - A performance comparison of kernel-level fibers wrt ULTs implemented in userspace
  - The benchmark will be provided on the course website soon

Some Hints

- Spend more time on the design than the implementation
- Write elegant code!
- Develop on a virtual machine
- Test both on a virtual machine and a physical machine
- Watch out interrupts and signals!

- If you need help: come to office hours!