Agenda for today

Hochschule RheinMain

- Groups!
- Any remaining problems for M0?

Feedback: some questions (1)

Hochschule RheinMain

- Describe the event paradigm, where is it typically used, what are its pros and cons?
- Why does it require only one stack for any number of possible events?
- In an event based kernel (i.e. distinct kernel / user spaces), how many distinct kernel stacks are required?
- Would an event based kernel be preemptible? What about user threads running on top of such a kernel? Would they be preemptible? ,

Feedback: some questions (2)

Hochschule RheinMain

- Coroutines can yield(), yet they are not preemptive. Explain.
- In the coroutine approach, do we have to expect / consider race conditions?
- What signifies a critical section in a coroutine approach (i.e. what must be avoided in a c.s.)?
- What are *callee saved* registers? What defines them ?
- what are *green threads*? What is the effect of holding a lock in the context of green trheads?

Feedback: some questions (3)

Hochschule RheinMain

- What data items need to be kept in a TCB?
- Could the entire stack of a thread be kept in the TCB as well?
- Can you think of any disadvantages of doing so?
- What items are being saved/restored where upon a thread switch in OS/161?

Feedback: some questions (4)

Hochschule RheinMain

- Can threads be implemented in user space alone?
- What are the advantages and disadvantages of doing so?
- In contrast, what are the advantages / disadvantes of kernel level threads?
- How could a *continuation* be represented in C?

Feedback: some questions (5)

- What represents the state of a thread blocked in the kernel in a per-thread kernel stack process model?
- Is preemption of a kernel thread possible / allowed at any point in the kernel when using ...
  - a per-thread kernel stack process model?
  - a process model based on continuations?
  - a stateless kernel model?
- Which of these models use a single kernel stack?
- Explain the advantages and disadvantages of the single kernel stack approaches

Feedback: some questions (6)

Hochschule RheinMain

- Why are threads a bad idea (according to John Ousterhout) for most purposes?
- What would justify the use of threads nevertheless, i.e. which use case *can not* be handled by events?
- What are the problems typically encountered in thread programming as opposed to event-based programming?