Losos3

Adam Hraška
Part of initial design by:
Peter Piják, Lukáš Krížik

Goals

- Fulfill requirements of 2nd extended assignment
  - Improved VM area management
  - Multiple memory heap allocation algorithms
- Simplify experimenting with core algorithms
  - Scheduling, memory allocation
  - Add/change without modifying other code
  - Abstract algorithm logic

C++ Experience

- Pros
  - OOP, especially polymorphism
  - Deterministic destruction/RAII
- Cons
  - No exceptions => no STL

Scheduler - design

- ThreadMgr
  - Thread lifecycle – in charge of thread object cleanup
  - create, kill
- Scheduler
  - Manages *live* threads
  - sleep, wakeup
- Scheduling strategy
  - Manages *runnable* threads
  - getNext

RR scheduling strategy

```
void RoundRobinScheduler::insertRunnable(Thread *pthread){
    runnable_.pushBack(*pthread);
}

void RoundRobinScheduler::removeRunnable(Thread *pthread){
    runnable_.erase(*pthread);
}

void RoundRobinScheduler::getNext(Thread *&pnext, Duration &quantum){
    if(!runnable_.empty()){
        pnext = &runnable_.front();
        runnable_.popFront();
        runnable_.pushBack(*pnext);
        quantum = defaultQuantum;
    }
    else{
        pnext = 0;
    }
}
```

Virtual memory map

- VM areas stored in red-black tree
  - Starting virtual address
  - Size
  - List of frames
- Free safety pages between areas
- VMM used to translate VA->PA; dedicated translator in future

Memory heap

- Same code for kernel and user-space
- Free blocks managed by allocation strategy
  - Accounting info within free block
- Block structure:
  - Header 4B – size, free/used flag
  - Footer 4B – size
- Debug heap
  - File name, line number, function of (de)allocation

Frame allocator

- Similar to memory heap – frames instead of bytes
- Memory heap allocation strategies used to manage contiguous free frames

Q & A

adam.hraska@yahoo.com