OSy group MiSmSv

Mário Mikula
Robert Smetana
Michal Švirec

Teamwork

- We had team before signing up for OSy
- Leader is the most experienced in C
- Communication by irc, jabber
- Sessions at school

- Sources: Kalisto, HelenOS, lectures

Threads

- Pointers stored in AVL tree
- Only two queues for threads
  - Running threads (scheduled)
  - Detached threads, unallocated on reschedule
- FIFO scheduling algorithm
  - Uses timers

Timers

- Priority queue for active timers
- Invoked timers queue
  - Multiple timers can be invoked in one interrupt
- Timer interrupts handled in special handler thread
  - Remove timer from invoked queue
  - Execute callback method
  - Everything with enabled interrupts

Frame allocator

- Bitmap of entire physical memory
  - Both mapped and unmapped
  - Write checked on startup
- Kernel heap allocated from the "front"
  - Continuous block, can be enlarged/shrunk
  - Next fit strategy
- Everything else allocated from the "end"
  - First fit strategy

Kernel heap

- Continuous in physical memory
- Constant initial size
- Size increases by constant when needed
- Decreases only from the tail
  - Not ideal in special case

Memory management

- Initially we implemented one level paging
  - Every thread used > 4MB of memory
  - Map2 tests needed > 20MB ☺
- Implemented 2 level paging
  - Memory requirements down to 280k
  - Realized the power of multilevel paging

Virtual memory allocator

- First fit strategy
- Sorted array of areas by start address

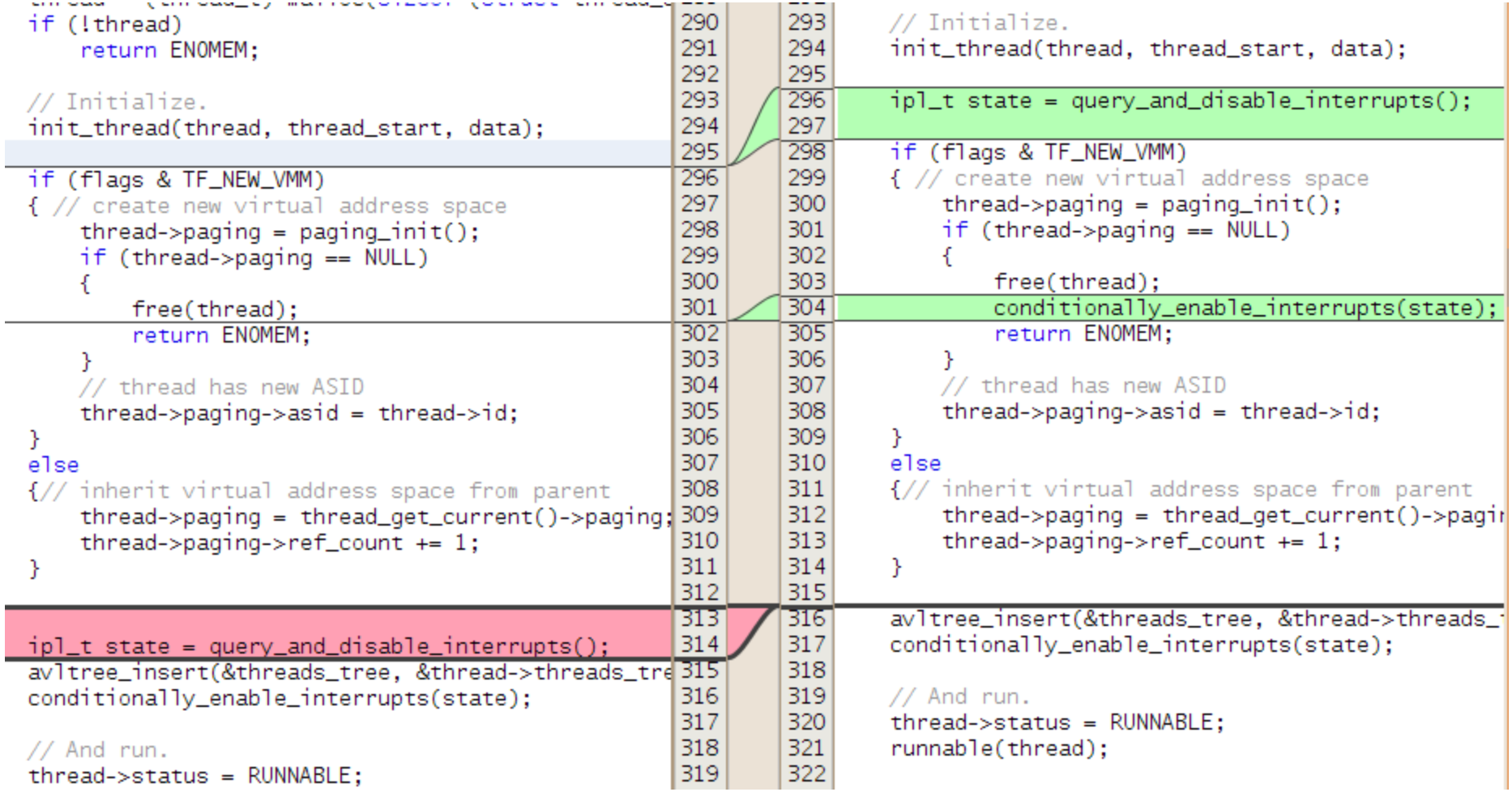
- Not a final version
- We are working on 3rd assignment

Next assignments

- Fixed number of registers for syscalls
  - Syscall number
  - 4 regs. for parameters
- Buffering for functions that output text

Deadly traps

- Don't allow timers only a few μs apart
- Double check values read/written to CP0 regs.
- Reserve more time for debugging
- Disable and enable interrupts on proper place



```
if (!thread)                                                    290 293     // Initialize.
    return ENOMEM;                                              291 294     init_thread(thread, thread_start, data);
                                                                292 295
// Initialize.                                                  293 296     ipl_t state = query_and_disable_interrupts();
init_thread(thread, thread_start, data);                        294 297
                                                                295 298     if (flags & TF_NEW_VMM)
if (flags & TF_NEW_VMM)                                         296 299     { // create new virtual address space
{ // create new virtual address space                           297 300         thread->paging = paging_init();
    thread->paging = paging_init();                             298 301         if (thread->paging == NULL)
    if (thread->paging == NULL)                                 299 302         {
    {                                                           300 303             free(thread);
        free(thread);                                           301 304             conditionally_enable_interrupts(state);
        return ENOMEM;                                          302 305             return ENOMEM;
    }                                                           303 306         }
    // thread has new ASID                                      304 307         // thread has new ASID
    thread->paging->asid = thread->id;                          305 308         thread->paging->asid = thread->id;
}                                                               306 309     }
else                                                            307 310     else
{// inherit virtual address space from parent                  308 311     {// inherit virtual address space from parent
    thread->paging = thread_get_current()->paging;              309 312         thread->paging = thread_get_current()->pagi
    thread->paging->ref_count += 1;                             310 313         thread->paging->ref_count += 1;
}                                                               311 314     }
                                                                312 315
                                                                313 316     avltree_insert(&threads_tree, &thread->threads_
ipl_t state = query_and_disable_interrupts();                   314 317     conditionally_enable_interrupts(state);
avltree_insert(&threads_tree, &thread->threads_tre              315 318
conditionally_enable_interrupts(state);                         316 319     // And run.
                                                                317 320     thread->status = RUNNABLE;
// And run.                                                     318 321     runnable(thread);
thread->status = RUNNABLE;                                      319 322
```

Thank You