12. NOVA Multithreading

Podpora více vláken v OS NOVA

Cílem tohoto cvičení je implementovat do OS NOVA systémová volání `thread_create` a `thread_yield` a dosáhnout tím podpory jednoduchých vícevláknových aplikací.

Domácí příprava

Pro toto cvičení budete potřebovat:

- přečíst si (ideálně i umět vysvětlit) funkce v OS NOVA:
  - `bootstrap`, `make_current`, `ret_user_sysexit`, konstruktor `Ec::Ec`
- mít základní znalosti o fungování plánovačů v operačním systému (schedulers) – viz 3. přednášku
- rozumět instrukci sysenter
- znát základní syntaxi inline assembleru (tak jako v minulých cvičeních)
  - https://gcc.gnu.org/onlinedocs/gcc/Extended-Asm.html (pochopit příklad z remarks)
- umět implementovat spojové seznamy
  - https://en.wikipedia.org/wiki/Linked_list
- základy programování v C++.

Zadání úlohy

> Tato úloha není povinná.

Implementujte systémová volání s následujícími prototypy:

```
int thread_create(void *(*start_routine)(void *), void *stack_top);
void thread_yield(void);
```

Požadavky

ABI

- `thread_create eax = 4, esi = start_routine, edi = stack_top`
- `thread_yield eax = 5`

Očekávané chování

- `thread_create` vytvoří vlákno a přidá ho do kruhové fronty vytvořených vláken.
- `start_routine` je ukazatel na adresu, od které se zahájí vykonávání nového vlákna.
- `stack_top` je adresa vrcholu zásobníku nového vlákna.
- `thread_create` vrací 0 pokud funkce proběhla bez chyby. V opačném případě vrací libovolné nenulové číslo chyby.
- `thread_yield` přepne vykonávání z aktuálního vlákna na následující vlákno ve frontě.
- Žádné ze systémových volání nesmí končit pádem systému.
- Postupné volání `thread_yield` přepíná mezi všemi vlákny ve frontě (žádné vlákno se nesmí vynechat).
- Běh vlákna se dá přerušit pouze voláním `thread_yield`.
- Pokud je zavoláno `thread_yield` po vytvoření nového vlákna, přepne `thread_yield` na naposledy vytvořené vlákno. Při dalším volání `thread_yield` se spustí vlákno, které by bylo spuštěno, pokud by se nové vlákno nevytvořilo.
- Za správné umístění a alokaci zásobníku pro nové vlákno je zodpovědný uživatelský program, který volá `thread_create`.

Co se odevzdává

- Změněný soubor `kern/src/ec_syscall.cc` implementující dvě nová systémová volání (implementace volání `brk` z 10. cvičení tam být nemusí). Archiv můžete vytvořit například pravidlem `hw10` v kořenovém souboru `Makefile` ve zdrojových kódech OS NOVA.

Materiály

- NOVA system call internals
- Ukázkový testovací program:

```
#include <stdio.h>

static inline unsigned syscall1 (unsigned w0)
{
  asm volatile (
      "   mov %%esp, %%ecx    ;"
      "   mov $1f, %%edx      ;"
      "   sysenter            ;"
      "1:                     ;"
      : "+a" (w0) :  : "ecx", "edx", "memory");
  return w0;
}

static inline unsigned syscall3 (unsigned w0, unsigned w1, unsigned w2)
{
  asm volatile (
      "   mov %%esp, %%ecx    ;"
      "   mov $1f, %%edx      ;"
      "   sysenter            ;"
      "1:                     ;"
      : "+a" (w0) : "S" (w1), "D" (w2): "ecx", "edx", "memory");
  return w0;
}

int create_thread(void (*start_routine)(void), void * stack)
{
    return syscall3(4, (unsigned long)start_routine, (unsigned long)stack);
}

void thread_yield(void)
{
    syscall1(5);
}

char stack[0x1000] __attribute__((aligned(16)));

void thread1()
{
    while (1) {
            printf("%s running\n", __func__);
```

```
            thread_yield();
        }
    }

    int main()
    {
      printf("Main thread running\n");

      create_thread(thread1, stack + sizeof(stack));
      while (1) {
            thread_yield();
            printf("I'm back in the main thread\n");
      }
    }
```