Laborator 6

Acțiuni amânabile

Sisteme de Operare 2 (SO2)

Departamentul de Calculatoare

Cuprins

Noțiuni generale

- motivația folosirii acțiunilor amânabile:
  - planificarea execuției unor acțiuni în viitor
  - scurtarea timpului de execuție al întreruperilor
  - în context întrerupere nu se pot folosi apeluri blocante
- softirq-uri, taskleti, timere, DPC-uri, workqueue-uri
- thread-urile kernel completează mecanismele de acțiuni amânabile
- au asociate o rutină de tratare (un handler)
- operații posibile asupra acțiunilor amânabile:
  - **inițializare:** stabilește rutina de tratare (handler-ul)
  - **planificare:** planifică execuția handler-ului (imediat sau la un anumit moment de timp)
  - **mascare:** dezactivare temporară

Cuprins

- nu pot fi folosite în drivere – sunt rezervate pentru subsistemele kernel-ului
- rulează în context întrerupere
- kernelul Linux 3.13 definește 10 astfel de acțiuni
- fiecare categorie are o prioritate asociată
- exemple:
  - `HI_SOFTIRQ` - rularea taskletilor prioritari
  - `TIMER_SOFTIRQ` - rularea timerelor
  - `TASKLET_SOFTIRQ` - rularea taskletilor
  - `HRTIMER_SOFTIRQ` - implementarea timerelor de mare precizie

Taskleti

- execută un handler la un moment ulterior de timp
- rulează în context întrerupere, fiind planificate peste softirq-uri
- nu se pot folosi apeluri blocante în handler
- declarare: `struct tasklet_struct`, `DECLARE_TASKLET`, `DECLARE_TASKLET_DISABLED`
- inițializare: `tasklet_init(&tasklet, handler, data)`
- planificare: `tasklet_schedule(&tasklet)`
- planificare cu prioritate: `tasklet_hi_schedule(&tasklet)`
- mascare: `tasklet_enable`, `tasklet_disable`

Timere

- execută un handler la un anumit moment de timp (în viitor, specificat în `jiffies`)
- rulează în context întrerupere, fiind planificate peste softirq-uri
- nu se pot folosi apeluri blocante în handler
- declarare: `struct timer_list`
- inițializare: `setup_timer(&timer, handler, data)`
- planificare: `mod_timer(&timer, expires)`
- oprire: `del_timer`, `del_timer_sync`

Locking softirq-uri

- mascare: `local_bh_disable` / `local_bh_enable`
- mascarea se aplică doar pe procesorul local
- mascarea acțiunilor amânabile poate fi imbricată
- spinlock-uri + mascare softirq-uri: `spin_lock_bh`, `spin_unlock_bh`

Workqueues

- planificare acțiuni care să ruleze în context proces
- operează cu unități numite **work**, care sunt de două tipuri:
  - `struct work_struct` – sarcină rulată la un moment ulterior
  - `struct delayed_work` – sarcină întârziată (pornește după cel puțin un interval de timp)
- declarare: `DECLARE_WORK` `DECLARE_DELAYED_WORK`
- inițializare: `INIT_WORK` `INIT_DELAYED_WORK`
- planificare: `schedule_work(&work)`
  `schedule_delayed_work(&work, delay)`
- anulare sarcini întârziate: `cancel_delayed_work(&work)`
- așteptarea terminării sarcinilor: `flush_scheduled_work()`
- există un workqueue implicit: `events`
- lucrul cu workqueue-uri: `create_workqueue`, `create_singlethread_workqueue`, `flush_workqueue`, `destroy_workqueue`, `queue_work`, `queue_delayed_work`

Kernel threads

- thread ce rulează în contextul procesului init (exclusiv în kernel mode)
- nu are spațiu de adresă asociat
- nu se poate accesa spațiul de adresă utilizator (nici măcar cu `copy_from_user`, `copy_to_user`)
- nu se poate implementa cod busy-waiting
- se pot chema operații blocante
- se pot folosi spinlock-uri / semafoare
- creare / execuție: `kthread_create_on_node`, `wake_up_process`, `kthread_run`
- terminare: `do_exit` (din acel thread)

Cuprins

Rezumat

- acțiunile amânabile oferă posibilitatea de a planifica execuția de cod la momente ulterioare de timp
- sunt folosite pentru a complementa funcționalitatea întreruperilor, permițând execuția acțiunilor care durează mai mult timp în afara întreruperii
- pentru locking se folosesc spinlock-uri (cu dezactivarea acțiunilor amânabile)
- Linux: softirq-uri (taskleti, timere), workqueues, kernel threads

Cuvinte cheie

- softirq
- tasklet
- timer
- bottom-half handlers
- `jiffies`, `HZ`
- `spin_lock/unlock_bh`
- workqueue
- kernel thread