„Ha beérjük annyival, hogy elátkozzuk vagy dicsőítjük a technikát, akkor sohasem jutunk el lényegének a megragadásához.”
Martin Heidegger

MIKROVEZÉRLŐS RENDSZERFEJLESZTÉS

ChibiOS/RT

Threads: Többszálú alkalmazás

Zsupányi Krisztián

THREADS: TÖBBSZÁLÚ ALKALMAZÁS

SZÁLKEZELÉS, MINEK?

- Sok mindent kell csinálni,
- Hogy tud egyszerre sok minden futni ?

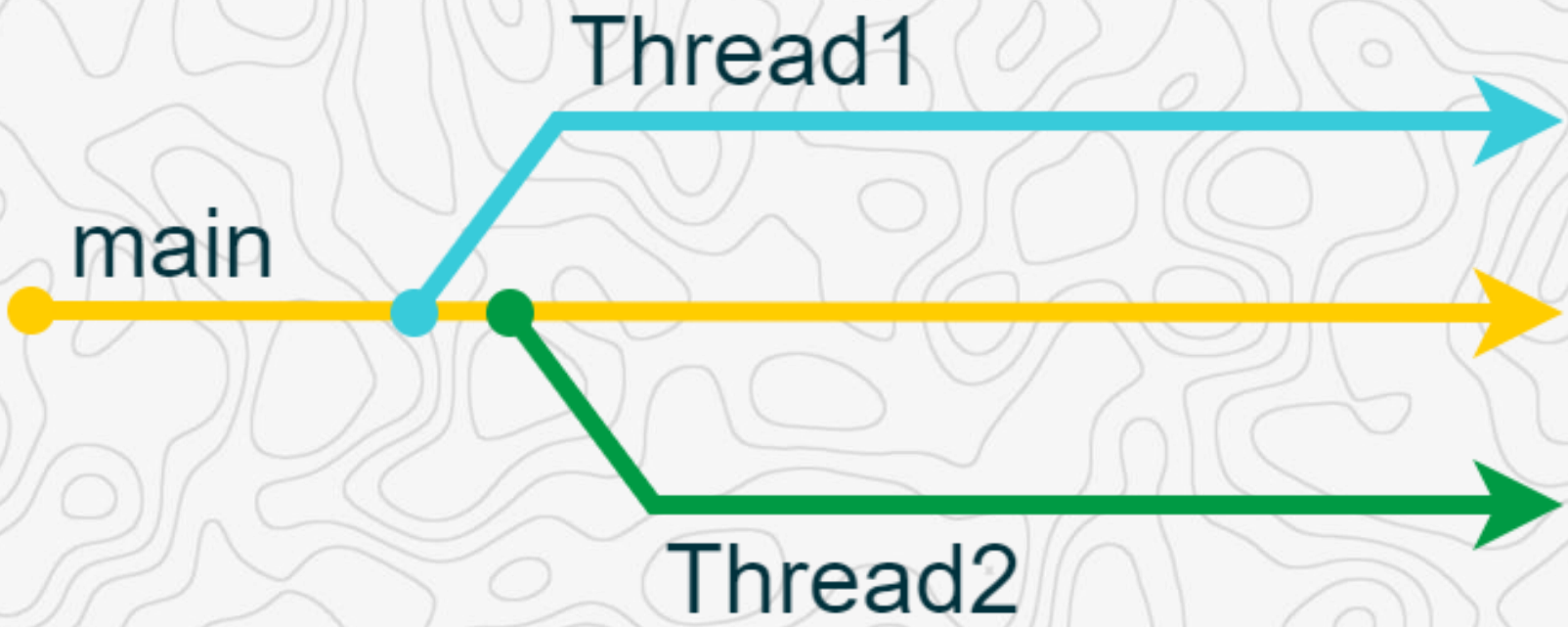
MI AZ A SZÁL? -> EGY NYÍL

A szál **egyetlen végrehajtási út**, egy utasítássorozat, amely általában egy **végtelen ciklusba lép a periodikus feladatok elvégzésére.**

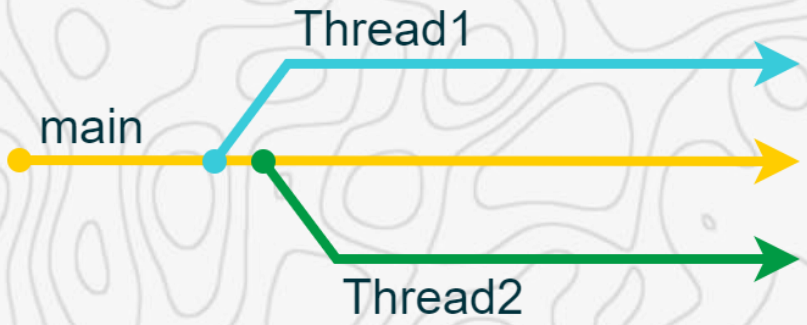
main

MULTI THREADING -> TÖBB NYÍL

De hát **egy magos processzoron**, bármely adott pillanatban valójában **csak egy szál fut**! **Akkor hogyan ?**

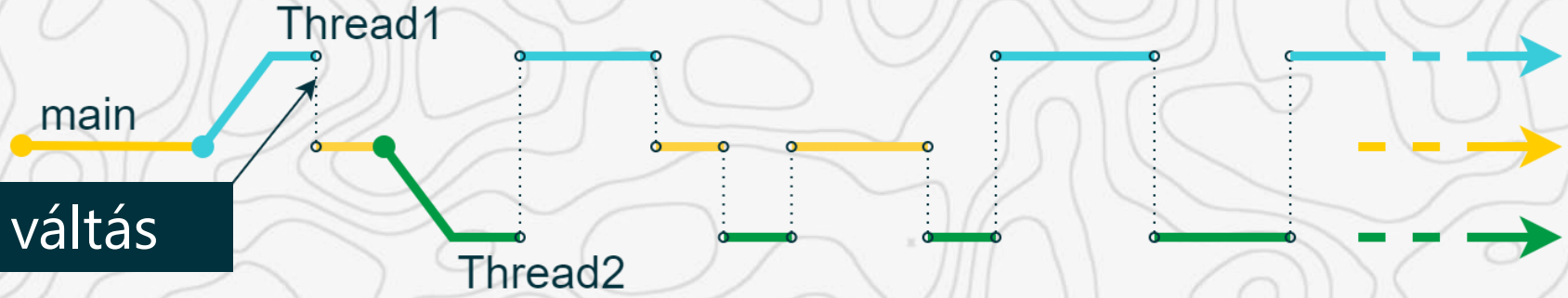


MULTI THREADING -> A VARÁZSLAT



Az operációs rendszer dönti el, hogy mikor melyik szál használhatja a CPU-t

Ideális eset
Valóságban



Kontextus váltás

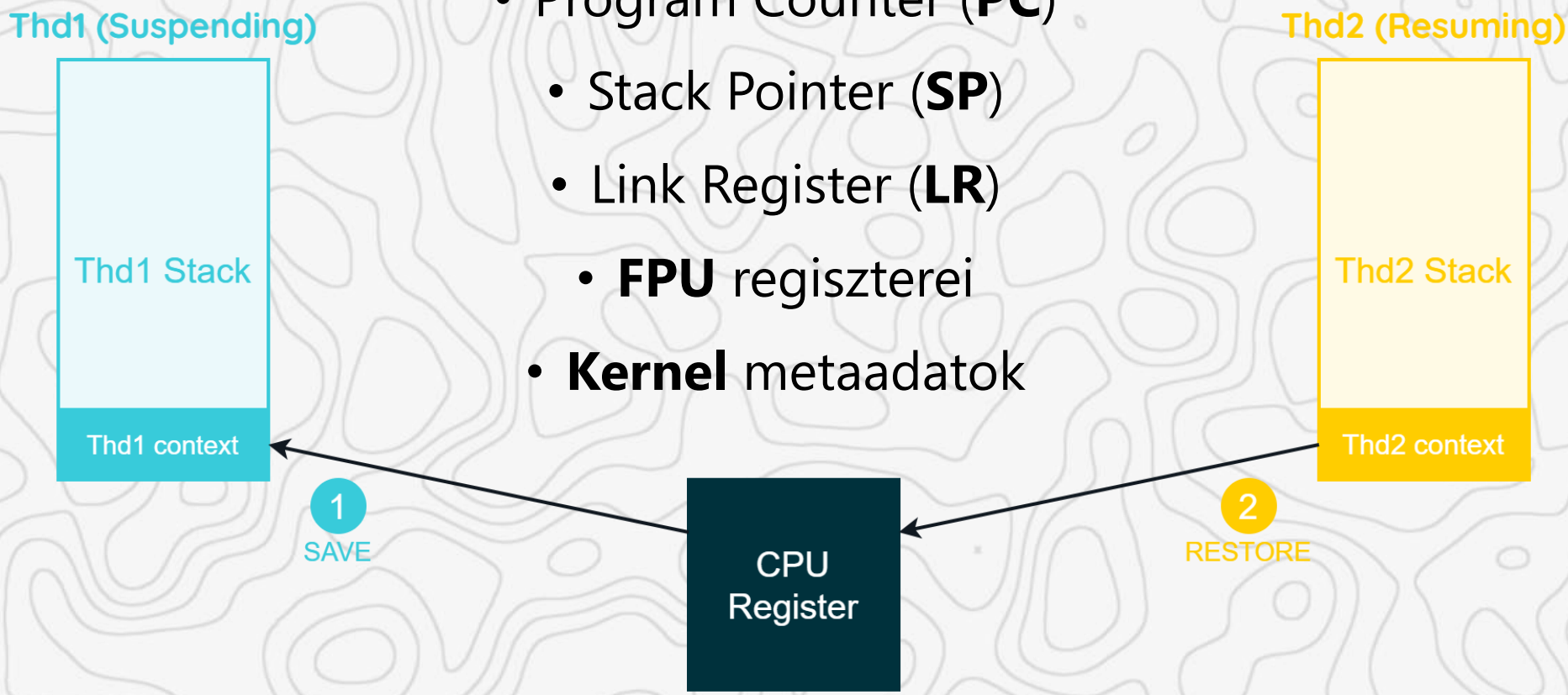
https://www.playembedded.org/blog/the-complete-reference-for-multithreading-in-chibios-rt/

Threads: Többszálú alkalmazás

KONTEXTUS VÁLTÁS

Futása megszakad, a CPU mindent elment. Hol tart a futásban:

- CPU általános regiszterek
- Program Counter (**PC**)
- Stack Pointer (**SP**)
- Link Register (**LR**)
- **FPU** regiszterei
- **Kernel** metaadatok



MI ALAPJÁN DÖNT AZ ÜTEMEZŐ

chThdWait(): Várakozás egy adott szál leállására.

Thread 1

chThdCreateStatic():
Statikus szál létrehozása és indítása.

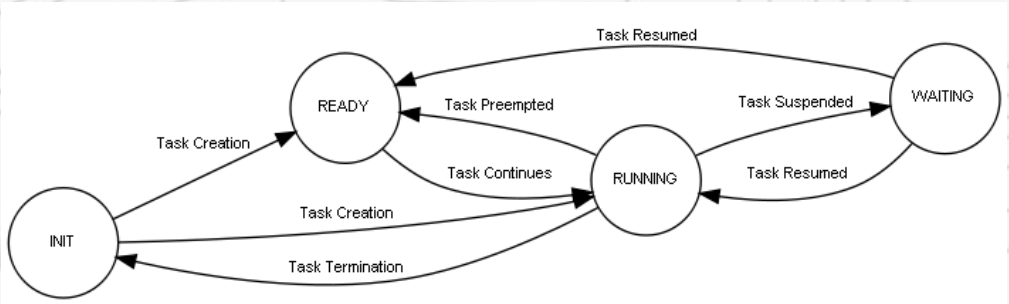
chThdExit():
Az aktuális szál leállítása.

Alapvető állapotok:

- **Fut**: Csak egy szál lehet ilyen, amelyiknek a **legmagasabb a prioritása a futásra készek közül**,
- **Vár**,
- **Felfüggesztett.**

Példák a felfüggesztett állapotokra:

- **CH_STATE_SLEEPING:** Alszik, időzítőre vár.
- **CH_STATE_WTMTX:** Mutexre vár.
- **CH_STATE_WTSEM:** Szemaforra vár.



PRIORITÁSI SZINTEK

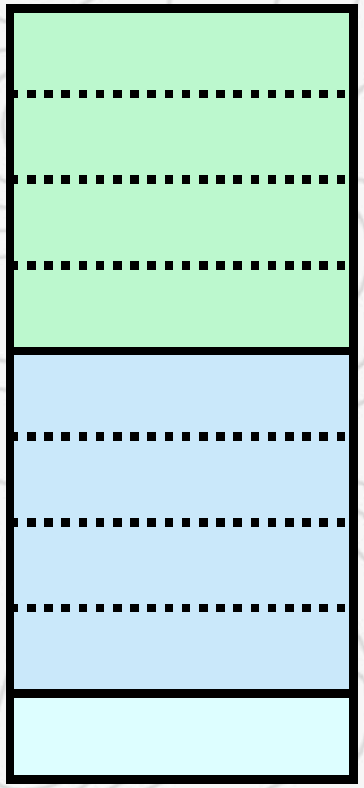
Maximum IRQ prioritás

Minimum IRQ prioritás

Maximum szál prioritás

Minimum szál prioritás

Idle szál prioritás



#define **HIGHPRIO** (tprio_t)**255**

#define **NORMALPRIO** (tprio_t)**128**

#define **LOWPRIO** (tprio_t)**2**

#define **IDLEPRIO** (tprio_t)**1**

#define **NOPRIO** (tprio_t)**0**

Megszakításoknak is van prioritása:

NVIC szabályozza, **ellentétes logika**

(kis szám = magas prioritás).

Magas prioritás: valós idejű, időkritikus feladatok (pl. motorvezérlés, gyors szenzor kezelés).

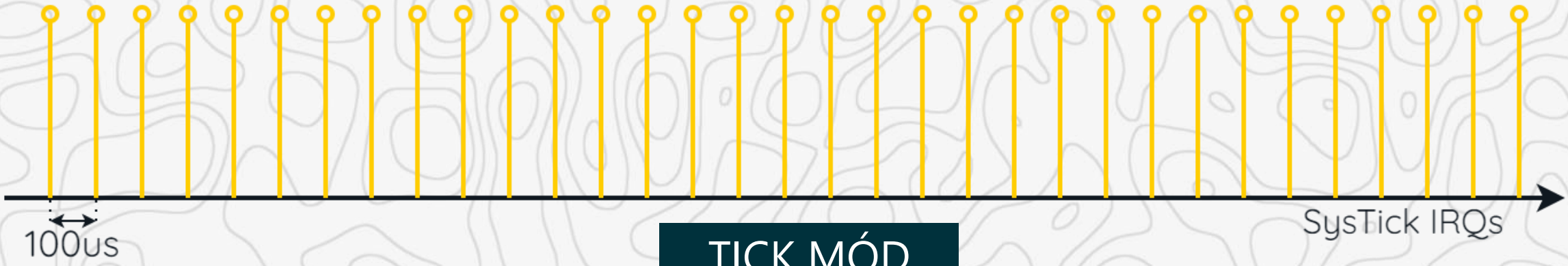
Közepes prioritás: feldolgozás, számítások.

Alacsony prioritás: háttérfeladatok (pl. logolás, LED villogtatás).

TICK MÓD ÉS TICKLESS MÓD

Egy **rendszeridőzítő (SysTick)** fix periódusban (pl. 1 ms) megszakítást generál az ütemezéshez.

CH_CFG_ST_TIMEDELTA = 0

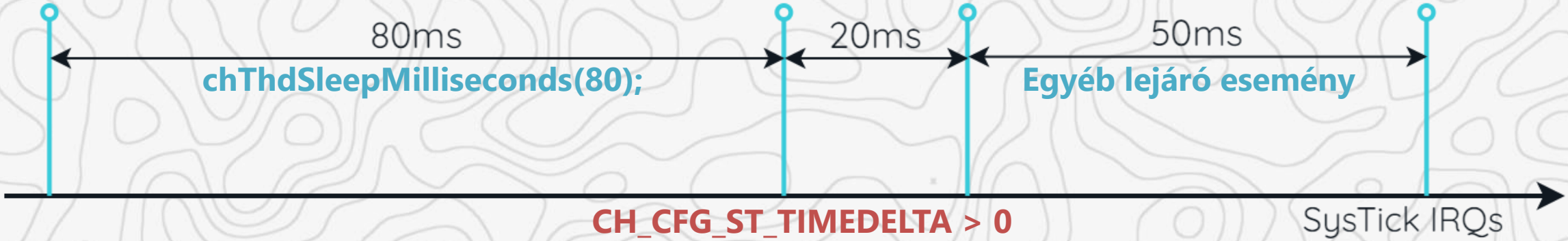


TICK MÓD

TICKLESS MÓD

Alapértelmezett

Energiahatékony



CH_CFG_ST_TIMEDELTA > 0

Nincs fix periódusú tick. Az RTOS a következő **lejáró eseményhez** programozza be a hardveres időzítőt.

Threads: Többszálú alkalmazás

PREEMPTÍV VS KOOPERATÍV ÜTEMEZÉS

CPU megosztás módja szálak között

Preemptív: magasabb prioritású szál azonnal megszakítja az alacsonyabbat → **mindig aktív ChibiOS-ben**

Kooperatív: azonos prioritású szálak csak önként (**chThdYield()** vagy **blokkolás**) adják át a CPU-t

Blokkoló pl: chEvtWait, spiExchange, adcConvert, i2cMasterTransmit, chSemWait, chThdSleep

A rendszer alap esetben **kooperatív** az **azonos prioritású szálak** között, de **preemptív** a **különböző prioritású** szálak között.

Azonos prioritású szálak esetén:

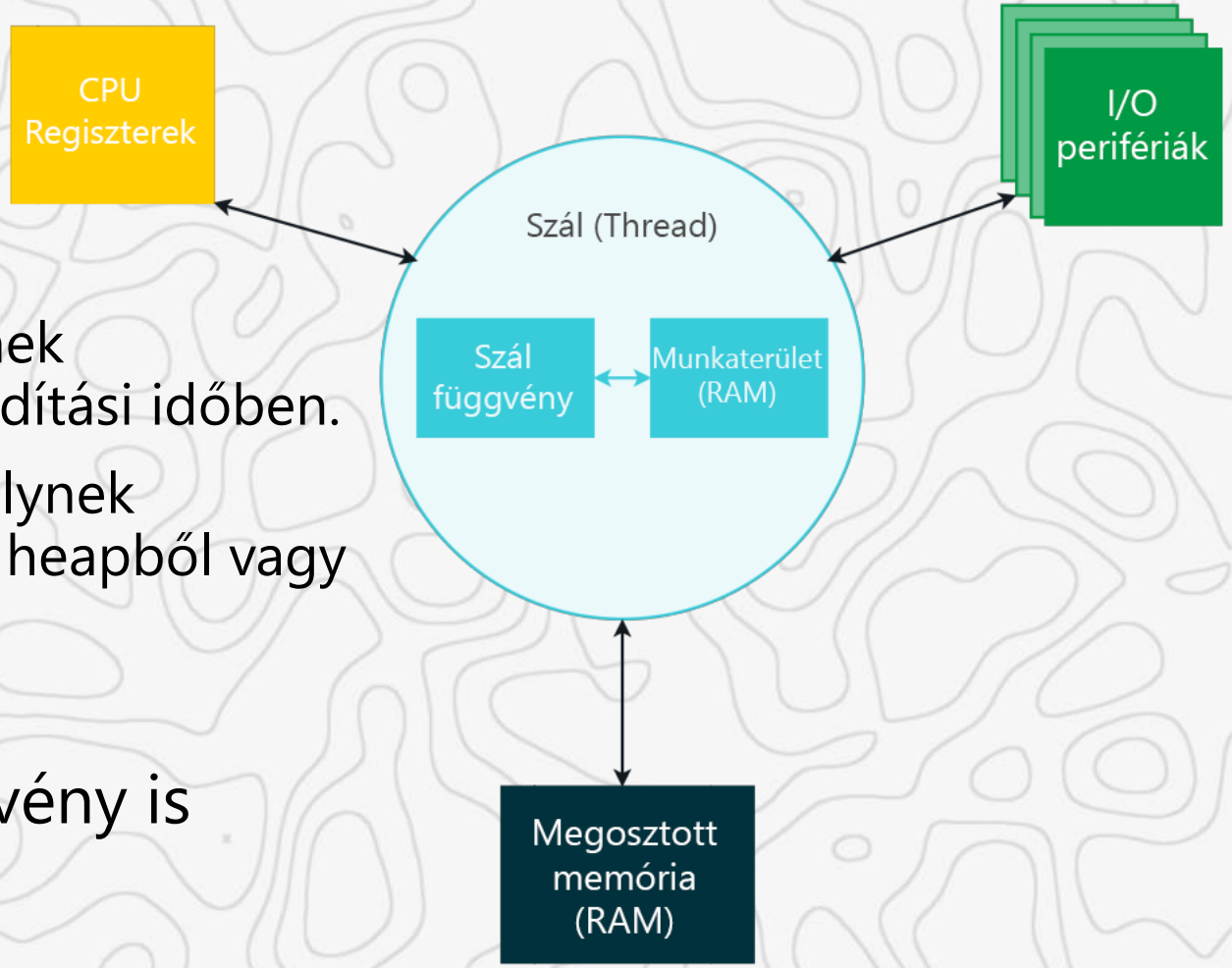
CH_CFG_TIME_QUANTUM = 0 → **Kooperatív**

CH_CFG_TIME_QUANTUM > 0 → **Round Robin**

A SZÁL (THREAD) ANATÓMIÁJA

Miből áll egy szál?

- Szál függvény (Thread function)
- Munkaterület (Working area)



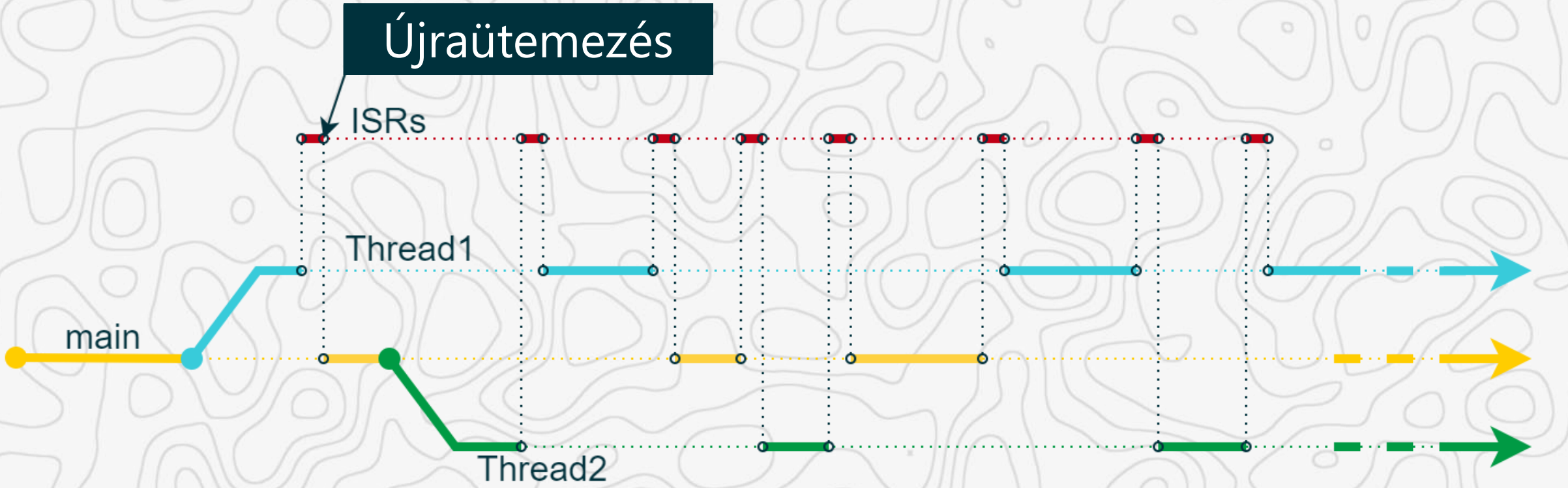
Statikus szál: Ez egy olyan szál, amelynek munkaterülete statikusan lefoglalt a fordítási időben.

Dinamikus szál: Ez egy olyan szál, amelynek munkaterülete futási időben lefoglalt a heapből vagy egy memóriakészletből.

Fő szál és Idle szál: A main függvény is szálként fut

ISR, INTERRUPT SERVICE ROUTINE

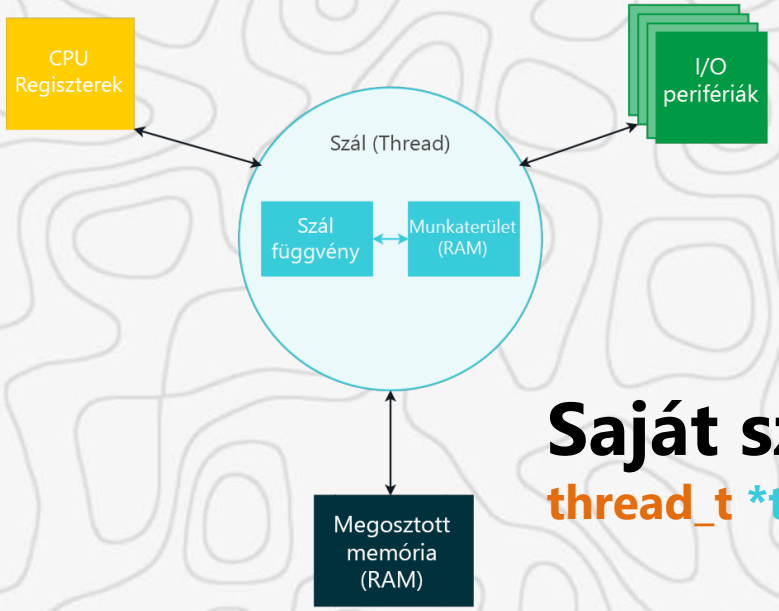
Mi áll a legnagyobb prioritású szál felett?



CHIBIOS: SZÁLAK ÉS SZINKRONIZÁCIÓ

Threads: Többszálú alkalmazás

SZÁL LÉTREHOZÁSA CHIBIOS-BEN



Saját szál mutató:

```
thread_t *tp = chThdGetSelfX();
```

Memória terület és függvény:

```
static THD_WORKING_AREA(waThread1, 128);
static THD_FUNCTION(Thread1, arg) {
    /* One-time setup for Thread1. */
    chRegSetThreadName("blinker");
    while (true) {
        /* Periodic task for Thread1. */
        ...
        chThdSleepMilliseconds(100);
        ...
    }
}
```

Statikus szál indítása:

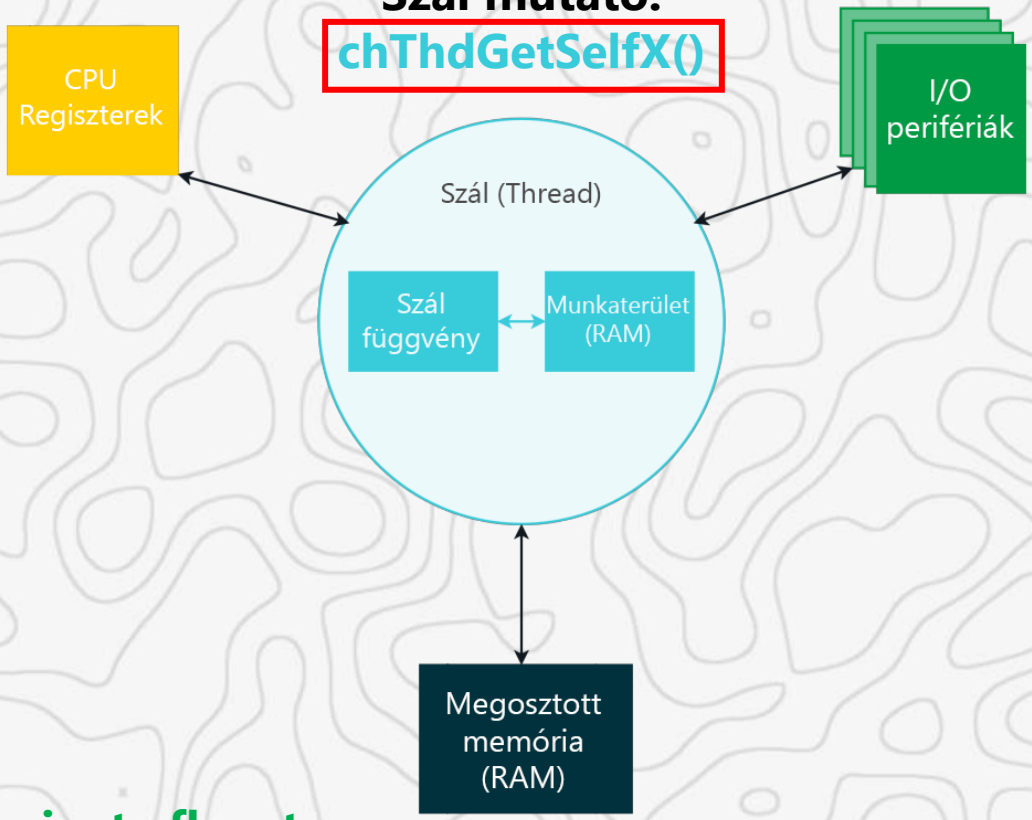
```
int main(void) {      //Dinamikus szálat mikrovezérlős környezetben nem szoktunk létrehozni
    ...
    thread_t *tp = chThdCreateStatic(waThread1, sizeof(waThread1), NORMALPRIO + 1,
    Thread1, NULL);
    ...
}
```

SZÁL LEÁLLÍTÁSA CHIBIOS-BEN

Szál leállítás kérésének vizsgálata:

```
static THD_WORKING_AREA(waThread1, 128);
static THD_FUNCTION(Thread1, arg) {
    /* One-time setup for Thread1. */
    // Your one-time code here
    while (!chThdShouldTerminateX()) {
        /* Periodic task for Thread1. */
        ...
        chThdSleepMilliseconds(100);
        ...
    }
}
```

Szál mutató:
chThdGetSelfX()



Statikus szál leállítása:

```
void stopBlink(void) {
    ...
    chThdTerminate(blinktp);    // beállítja a terminate flaget
    chThdWait(blinktp);         // megvárja míg a szál leáll
}
```

SYSLOCK

Megáll, mint a szög!

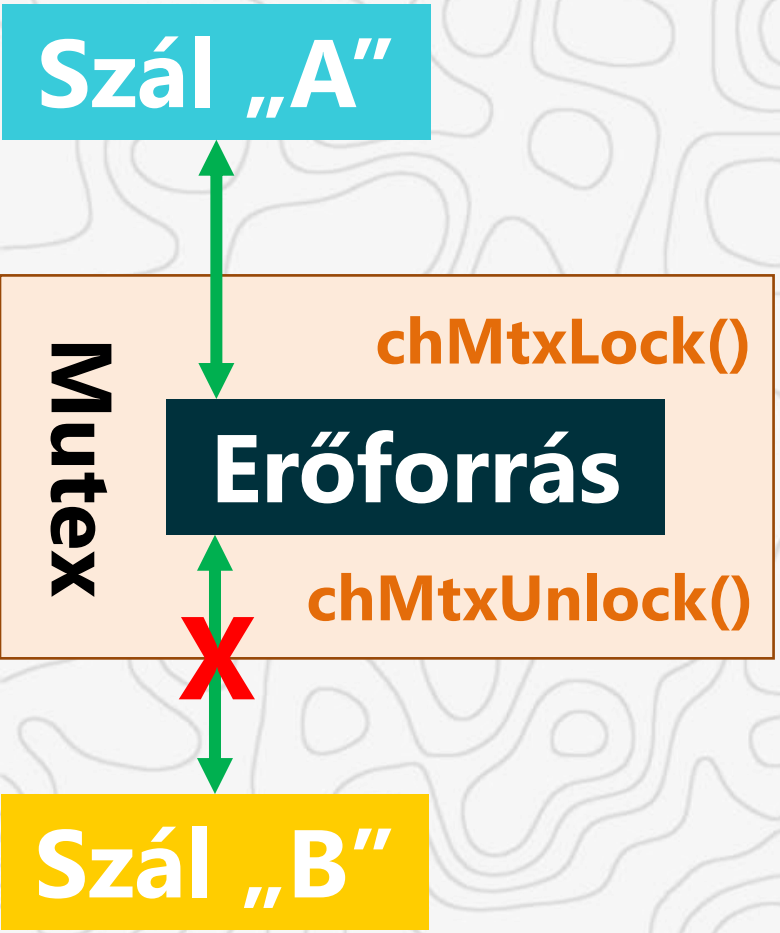
Mit tehetünk ha nem szeretnénk, hogy egy magasabb prioritású szál vagy ISR megszakítson egy kódot?

A végrehajtást ilyenkor **nem szakíthatja meg semmi**!

```
chSysLock ( void ) ;
/* A kritikus zóna, szál kontextusában. */
chSysUnlock ( void ) ;

chSysLockFromISR ( void ) ;
/* A kritikus zóna, ISR kontextusban. */
chSysUnlockFromISR ( void ) ;
```

SZINKRONIZÁCIÓ (MUTEX)



Mutex (Mutual Exclusion)

👉 **Cél:** közös erőforrás **kizárólagos védelme**

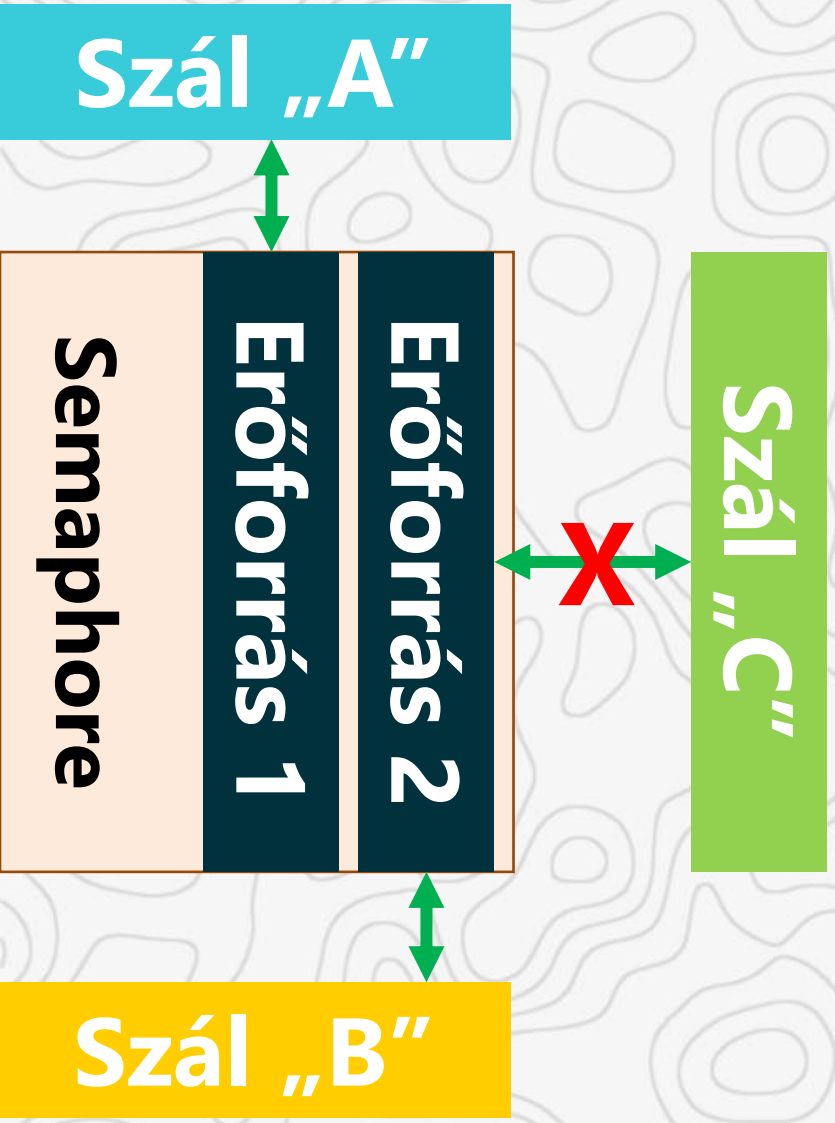
Egyszerre **csak egy szál tarthatja** a mutexet.

Tipikusan rövid időre foglaljuk (pl. egy soros porthoz írás).

Ha másik szál is szeretné használni → várakozik, amíg a mutex felszabadul.

Prioritás öröklés: ha egy alacsony prioritású szál tartja a mutexet, és egy magas prioritású szál vár rá → ideiglenesen örökli a magasabb prioritást → **megelőzi a *priority inversion*** problémát.

SZINKRONIZÁCIÓ (SZEMAFOR)



Semaphore (számláló, van bináris is)

👉 **Cél:** korlátozott számú erőforrás kezelése

Számláló szemafor: pl. van 3 azonos erőforrás → számláló = 3. Minden foglalás csökkenti, felszabadítás növeli.

Nem tartalmaz prioritás öröklést.

```
chSemObjectInit(&uartsem, UART_CHANNELS)
chSemSignal(&uartsem)
chSemWait(&uartsem)
```

N < 0. A szemafor foglalt, és **N szál van sorban**.

N == 0. A szemafor foglalt, de **nincsenek szálak sorban**.

N > 0. A szemafor nem foglalt, és **N alkalommal vehető fel.**

SZINKRONIZÁCIÓ (MAILBOX)

👉 **Cél:** adatküldés és fogadás szálak között

A **termelő (producer)** szál adatot küld → **bekerül a sorba.**

A **fogyasztó (consumer)** szál kiveszi → **feldolgozza.**

FIFO elv (elsőként betett elsőként jön ki).

SZINKRONIZÁCIÓ (ESEMÉNYEK)

👉 **Cél:** szálak közötti esemény jelzés, akár többféle esemény egy szálhoz.

Bitmask alapon működik → egy szál több eseményre is várhat. Például:

- 0x01 → új adat érkezett soros portra
- 0x02 → időzítő lejárt
- 0x04 → motor hibát jelzett

Esemény regisztrálása: `chEvtRegisterMask(souce, listener, EVENT_MASK(1))`

Blokkoló várakozás eseményre: `chEvtWaitAny(ALL_EVENTS)` → a szál addig blokkol, amíg valamelyik esemény be nem következik.

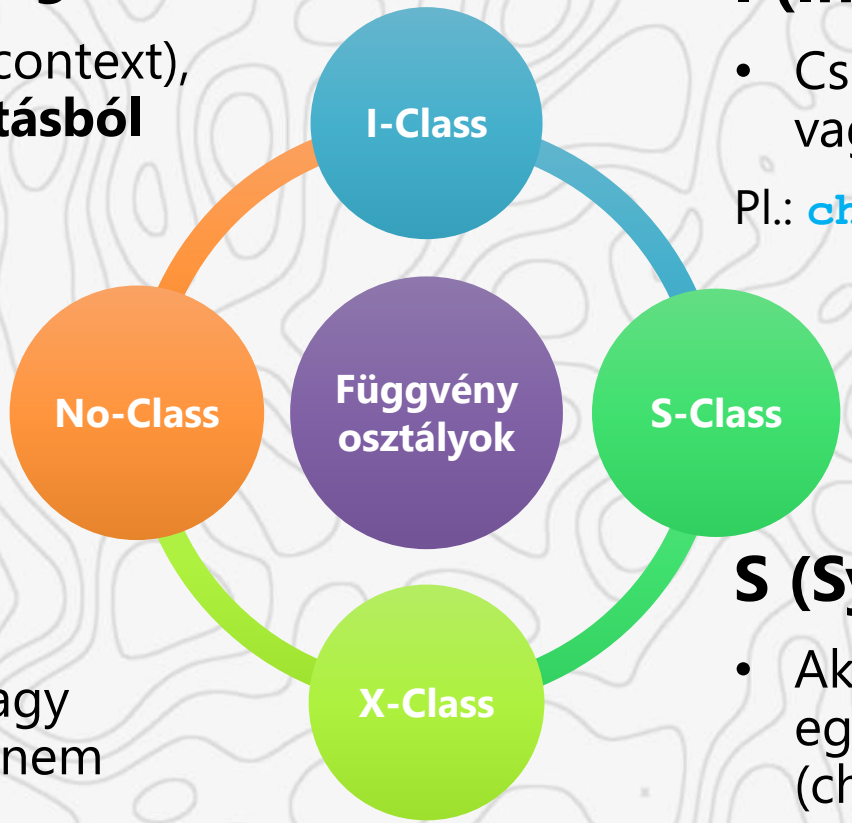
Esemény típusának megállapítása: `if(evt & EVENT_MASK(1)) ...`

FÜGGVÉNY OSZTÁLYOK

Normál függvények (nincs végződés)

- Csak szálból hívhatók (thread context), **nem használhatók megszakításból vagy kritikus szekcióból**,

Pl.: `chThdSleepMilliseconds()`



I (Interrupt safe)

- Csak megszakítási kontextusból vagy lockolt rendszerből hívhatók,

Pl.: `chSemSignalI()`

X (Critical section)

- Biztonságos megszakításból vagy időkritikus szakaszból is, mert nem ragadhat be a szál,

Pl.: `chThdShouldTerminateX()`

S (System locked)

- Akkor használjuk, ha már beléptünk egy system lock szakaszba (chSysLock()),
- Pl.: `chSemWaitS()`

„Ha beérjük annyival, hogy elátkozzuk vagy dicsőítjük a technikát, akkor sohasem jutunk el lényegének a megragadásához.”

Martin Heidegger

KÖSZÖNÖM A FIGYELMET!

Zsupányi Krisztián

Threads: Többszálú alkalmazás