

Mutex, LockGuard i inheritance

owner, trzy early returns i zmiana LOW 1–3–1

Karta	K11 / 16	Czas	30 minut
Platforma	Hazard3 / RV32I	Taski	LOW/MEDIUM/HIGH
Priorytety	1 / 2 / 3	Storage	static, heap delta 0
Wersja	v00.01	UUID	34d7ad61-...
		karty	

Dlaczego mutex nie jest binary semaphore?

Właściwość	Mutex	Binary semaphore
owner	tak	nie
give	tylko owner	sygnał od producenta
priority inheritance	tak	nie
ISR give	nie	wariant FromISR istnieje

Kontrakt K11. HIGH blokuje się na mutexie LOW. Kernel tymczasowo podnosi LOW do priorytetu HIGH, aby MEDIUM nie przedłużał inwersji. Destruktor guarda zwalnia dokładnie raz; ordinary mutex nie jest rekurencyjny ani ISR-safe.

Plan 30 minut

Czas	Tryb	Dowód
0–5	semantics	owner, give i różnica od semaphore
5–10	guard	trzy return, jeden unlock na ścieżkę
10–15	setup	LOW bierze; HIGH próbuje; MEDIUM ready
15–20	inherit	HIGH blocked; LOW 1–3
20–25	handoff	LOW przed MEDIUM; potem HIGH owner
25–30	restore	LOW wraca do 1; counter=11

Predykcja

Kto wykona się po zablokowaniu HIGH: LOW czy MEDIUM? _____. Jaki będzie bieżący priorytet LOW? _____.

1 LockGuard i granice ownership

```
{
    freertos::LockGuard<freertos::Mutex> guard(mutex, timeout);
    if (!guard.owns_lock()) return failed;
    // protected work; every return crosses guard destructor
}
```

Path	Take	Give po return	Wynik
0	success	_____	10
1	success	_____	20
2	success	_____	30

Ordinary oznacza non-recursive

Ten sam task próbuje wziąć już posiadany mutex z timeout 0. Wynik: _____. Czy nieudany LockGuard może wywołać give w destruktorze? _____.

2 Kontrolowany scenariusz inwersji

LOW(1) take - HIGH(3) attempt/block - LOW inherits 3
LOW release - HIGH take/give - MEDIUM(2) run - LOW back at 1

Nr	Aktor	Zdarzenie	Odczyt
1	LOW	take, owner=LOW	_____
2	HIGH	attempt, state=Blocked	_____
3	LOW	priority inherited, release	_____
4	HIGH	take, owner=HIGH	_____
5	MEDIUM	pierwszy run	_____
6	LOW	priority restored	_____

Dlaczego MEDIUM nie może znaleźć się między krokiem 2 i release LOW?

3 Dowód priority inheritance

Pomiar	Predykcja	Odczyt
LOW base priority	1	_____
HIGH priority	3	_____
HIGH state podczas wait	Blocked	_____
LOW current podczas wait	3	_____
MEDIUM ran before release	0	_____
LOW po release	1	_____

4 Owner i chroniony stan

Relacja	Predykcja	Odczyt
holder po LOW take	LOW handle	_____
holder po handoff	HIGH handle	_____
counter po LOW	1	_____
counter po HIGH	11	_____
mutex handle	caller control	_____
heap delta create/destroy	0 / 0	_____

Co inheritance rozwiązuje, a czego nie?

Rozwiązuje ograniczoną inwersję powodowaną przez ownera mutexa. Nie skraca długiej sekcji, nie naprawia deadlocku, nie nadaje porządku wielu mutexom i nie pozwala zwolnić mutexa z innego taska.

Zaproponuj stały porządek przejmowania dwóch mutexów A/B:

Co stanie się z inheritance, gdy LOW trzyma kilka mutexów?

Kontrakt czasu

Timeout w konstruktorze guarda jest jawny. `portMAX_DELAY` w tym eksperymencie jest świadomym oczekiwaniem HIGH, nie ukrytym defaultem API.

5 Hazard3/GDB i zaliczenie

```
make check
riscv64-unknown-elf-gdb build/task01_mutex/prog.elf
b mutex_debug_checkpoint
```

```
p g_low_base_priority
p g_low_inherited_priority
p g_low_priority_after_release
p g_high_state_seen_by_low
p g_medium_ran_before_release
p/x g_owner_low
p/x g_owner_high
p g_sequence_low_take
p g_sequence_high_attempt
p g_sequence_low_release
p g_sequence_high_take
p g_sequence_medium_run
p g_shared_counter
p g_mutex_pass
```

Zaliczenie

- ☐ odróżniam mutex owner/inheritance od binary semaphore;
- ☐ pokazuję give guarda na trzech early returns;
- ☐ dowodzę HIGH=Blocked i LOW 1-3-1;
- ☐ MEDIUM nie wykonuje się przed release LOW;
- ☐ owner równa się właściwemu handle LOW/HIGH;
- ☐ event order i chroniony counter=11 kończą się PASS.

Wyjście

Dlaczego binary semaphore nie naprawi tej inwersji?

Dlaczego ordinary mutex nie wolno brać drugi raz?

Następna karta K12: binary/counting semaphore oraz jednoodbiorcze task notifications z porównaniem storage i semantyki.