

Semaforey i task notifications

sygnał 0/1, licznik slotów i kanał jednego TCB

Karta	K12 / 16	Czas	30 minut
Platforma	Hazard3 / RV32I	Taski	waiter 3 / signaler 2
Storage	static controls + TCB	Heap delta	0 B
Wersja	v00.01	UUID	fcb6107c-...
		karty	

Trzy podobne wakeupy, trzy różne kontrakty

Typ	Stan	Odbiorcy	Payload
Binary semaphore	0 lub 1	waiter obiektu	brak
Counting semaphore	0..Max	waiterzy obiektu	count zasobu
Task notification	value/state w TCB	jeden task/slot	value lub bits

Kontrakt K12. Semaphore nie jest magazynem danych. Notification jest adresowana do konkretnego taska, a action oraz clear/decrement policy są częścią typu operacji, nie szczegółem do domyslenia.

Plan 30 minut

Czas	Tryb	Dowód
0–5	binary	0–1–0; drugi give przy full=false
5–10	counting	initial/max 2; take 2–1–0
10–15	block	trzeci take: waiter eBlocked
15–20	release	jeden give budzi; token od razu zużyty
20–25	notify count	TCB give/take zwraca 1
25–30	notify value	overwrite/wait daje 0xA5A55A5A

Wybór

Jeden producent budzi jeden znany task bez payloadu: _____. Trzech workerów konkuruje o dwa sloty: _____.

1 Binary i counting: granice count

Binary krok	Predykcja	Odczyt
po create	0	_____
po pierwszym give	1	_____
drugi give przy full	false / nadal 1	_____
po take(0)	0	_____

Counting krok	Predykcja	Odczyt
max / initial	2 / 2	_____
give przy max	false / 2	_____
po take 1	1	_____
po take 2	0	_____
take 3, portMAX delay	waiter Blocked	_____
signaler give / handoff	waiter wakes, count 0	_____

2 Dlaczego count po handoff nadal wynosi zero?

Give przekazuje dostępność oczekującemu waiterowi. Ponieważ waiter ma wyższy priorytet, natychmiast kończy zablockowany take i konsumuje token, zanim signaler ponownie odczyta count.

Narysuj kolejność: waiter block, signaler give, waiter ready/run/take, signaler resume.

Storage

Oba uchwytty muszą równać się caller `StaticSemaphore_t` w `.bss`. Create i destroy nie zmieniają heapu. Notification nie ma trzeciego control block: value/state istnieją już w TCB waitera.

3 Notification jako count 1

```
freertos::TaskNotification target{waiter_handle};
target.give();
// in waiter:
uint32_t n = freertos::TaskNotification::take(true, timeout);
```

Pomiar	Predykcja	Odczyt
waiter przed give	Blocked	_____
notification value po give	1 pending	checkpoint / GDB
take clear-on-exit result	1	_____
value po take	0	_____

4 Notification jako jawna wartość

```
target.notify(0xA5A55A5A,
              freertos::NotificationAction::overwrite);
uint32_t value;
TaskNotification::wait_value(value, timeout);
```

Pomiar	Predykcja	Odczyt
waiter przed notify	Blocked	_____
action	overwrite	_____
received value	0xA5A55A5A	_____
clear on exit	wszystkie bits	_____

Clear kontra decrement

`take(true)` zeruje cały count i zachowuje się jak binary. Z `take(false)` odejmuje jeden i zachowuje się jak counting. Co zwrócą dwa kolejne `take(false)`, gdy przed nimi wykonano trzy `give`?

Ćwiczenie wyboru

Dobierz mechanizm: (a) dwa wolne bufory DMA, (b) UART budzi konkretny parser, (c) przycisk publikuje strukturę z timestampem.

5 Hazard3/GDB i zaliczenie

```
make check
riscv64-unknown-elf-gdb build/task01_semaphore_notify/prog.elf
b synchronization_debug_checkpoint
```

```
p g_binary_initial_count
p g_binary_after_give
p g_binary_second_give
p g_counting_initial_count
p g_count_after_take1
p g_count_after_take2
p g_waiter_state_for_counting
p g_waiter_state_for_notification
p g_notification_pending_checkpoint
p g_notification_take_value
p/x g_notification_message_value
p g_sync_pass
```

Zaliczenie

- ☐ rozróżniam binary signal, counting slots i TCB notification;
- ☐ pokazuję binary 0–1–0 i odrzucony drugi give;
- ☐ pokazuję counting 2–1–0, Blocked i handoff;
- ☐ notification give/take zwraca dokładnie 1;
- ☐ overwrite/wait przenosi 0xA5A5A5A;
- ☐ wyjaśniam storage, cardinality i clear policy.

Wyjście

Dlaczego notification nie zastąpi kolejki wielu konsumentów?

Dlaczego semaphore count nie jest payloadem aplikacji?

Następna karta K13: typed EventBits, wait any/all, clear-on-exit oraz koordynacja stanu wielu tasków.