Struktura jąder systemów operacyjnych

Lista zadań nr 3

Na zajęcia 1 kwietnia 2020

UWAGA! W trakcie prezentacji rozwiązań należy zdefiniować i wyjaśnić pojęcia, które zostały oznaczone **wytłuszczoną** czcionką.

Autor kodu przyznaje się do tego, że omawiane niżej sterowniki nie obsługują wszystkich przypadków brzegowych, nie są wolne od błędów oraz nie posiadają pięknej dokumentacji. Osoby chętne do wprowadzenia poprawek do kodu mogą uzyskać punkty bonusowe po uwcześniejszym ustaleniu zakresu prac.

Zadanie 1. Przeprowadź słuchaczy zajęć przez źródła prostego sterownika portu szeregowego serial.c[1]. Rozpocznij od procedury inicjującej «SerialInit» i wyłączającej sterownik «SerialKill». Skup się na omówieniu komunikacji między dolną i górną połówką przy pomocy **kolejek** systemu FreeRTOS. Do czego służy procedura «TriggerSend»? Jakiej długości są sprzętowe bufory odbiornika i nadajnika, a jakiej programowe? Czemu priorytety przerwań RBF (ang. *receiver buffer full*) i TBE (ang. *transmit buffer empty*) są różne? Opis sprzętu można znaleźć w rozdziale Serial I/O Interface[2].

Zadanie 2. Przeprowadź słuchaczy zajęć przez źródła czasomierza cia-line.c[3] zliczającego linie rastra. Raster to jedna klatka obrazu w standardzie PAL[4] składająca się z 312 linii, z których każda trwa 64μs. Skup się na omówieniu komunikacji między dolną i górną połówką przy pomocy **powiadomień** systemu FreeRTOS. Zauważ, że z czasomierza może korzystać wiele zadań równocześnie – jak zostało to osiągnięte? Opis sprzętu można znaleźć w rozdziale 8520 Complex Interface Adapters[5].

Zadanie 3. Na podstawie przykładu «preemption» utwórz w katalogu «examples» nowy program o nazwie «rtc». Należy odpowiednio zmodyfikować plik «examples/Makefile». Program powinien utworzyć jedno zadanie, które będzie wybudzane co sekundę po to, by odczytać wartość z zegara czasu rzeczywistego i wydrukować ją w formacie «DD-MM-YY HH:MM:SS» przy użyciu procedury «printf» na port równoległy. Wiemy, że procedura «xTaskIncrementTick» jest wołana 50 razy na sekundę. Zegar czasu rzeczywistego to układ M6242B[6], którego zawartość jest widoczna dla programu przez strukturę «msm6242b» zdefiniowaną w pliku rtc.h[7].

Zadanie 4. Omów działanie **oddania sterowania** przez wywołanie instrukcji «trap #0», a zatem wykonanie «vPortYieldHandler». Czemu pole «pxTopOfStack» musi być pierwsze w strukturze «tskTCB»? Zreferuj działanie procedury «vTaskSwitchContext» ze szczególnym uwzględnieniem wyboru następnego zadania do wykonania. Możesz pominąć zarządzanie zmienną errno(2) i kod związany z biblioteką newlib.

Zadanie 5. W każdym z przykładów z katalogu «examples» procedura «main» kończy się wywołaniem procedury «vTaskStartScheduler», z której sterowanie z reguły już nie wraca. Co się dzieje w trakcie rozruchu planisty? Pokaż w jaki sposób procesor zaczyna wykonywać instrukcje pierwszego wybranego zadania. Kto i jak tworzy ustalany rozruchowy kontekst zadania?

Zadanie 6. Przeprowadź uczestników zajęć przez proces wysyłania i odbierania powiadomień zadań przy użyciu procedur «xTaskNotify» i «xTaskNotifyWait». W trakcie prezentacji skup się na pokazaniu kiedy zadania są **usypiane**, **wybudzane** lub **wywłaszczane**. W przypadku drugiej procedury pokaż w jaki sposób przetwarzane są argumenty «ulBitsToClearOnEntry» i «ulBitsToClearOnExit».

[1] https://mimiker.ii.uni.wroc.pl/source/xref/FreeRTOS-Amiga/drivers/serial.c
[2] http://amigadev.elowar.com/read/ADCD_2.1/Hardware_Manual_guide/node019F.html
[3] https://mimiker.ii.uni.wroc.pl/source/xref/FreeRTOS-Amiga/drivers/cia-line.c
[4] http://martin.hinner.info/vga/pal.html
[5] http://amigadev.elowar.com/read/ADCD_2.1/Hardware_Manual_guide/node012E.html
[6] https://www.amigawiki.org/lib/exe/fetch.php?media=de:parts:m6242b_oki_datasheet.pdf
[7] https://mimiker.ii.uni.wroc.pl/source/xref/FreeRTOS-Amiga/include/rtc.h

Zadanie 7. Zaprezentuj uczestnikom zajęć tworzenie zwykłego **muteksa** «`xSemaphoreCreateMutex`» oraz **semafora binarnego** «`xSemaphoreCreateBinary`» i wskaż różnice. Następnie przeprowadź ich przez proces zakładania i zwalniania muteksów przy użyciu procedur «`xSemaphoreTake`» i «`xSemaphoreGive`». Pokaż **sekcje krytyczne**, które wyłączają przerwania lub wywłaszczanie. W trakcie prezentacji skup się na wyjaśnieniu implementacji mechanizmu **dziedziczenia priorytetów** – pokaż kiedy zmieniany jest priorytet zadania, które jest właścicielem muteksa.