**RTAI**

(Real Time Application Interface)



Jakub Jachowicz

Informatyka w Sterowaniu i Zarządzaniu

Informatyka Czasu Rzeczywistego

Wstęp

RTAI to dodatek do systemu Linux, który jest udostępniony na zasadach open-source. Działa on na bazie “*sub-kernel*” i posiada twarde ograniczenia czasowe (tzw. “*hard real-time*”). Poza tymi samymi usługami, co zwykłe jądro Linuksa, RTAI dodatkowo udostępnia usługi powiązane z systemem czasu rzeczywistego.

System czasu rzeczywistego

System taki można zdefiniować jako system zdolny zagwarantować ograniczenia czasowe wykonania procesów, które na za zadanie wykonać.

System czasu rzeczywistego musi mieć dostatecznie małą latencję. Musi odpowiadać zewnętrznym asynchronicznym zdarzeniom w możliwie krótkim czasie w razie, gdy zdarzenia te potrzebują natychmiastowej odpowiedzi.

Zazwyczaj zadania czasu rzeczywistego mają postawione wymagania:

* czasowe: zadania można podzielić na periodyczne lub aperiodyczne; te pierwsze muszą być powtarzane co pewien okres czasu; te drugie mają odgórnie ustalony czas rozpoczęcia lub zakończenia zadania (lub zarówno początku, jak i zakończenia);
* dostępowe: zadania czasu rzeczywistego mogą potrzebować dostępu do baz danych, pamięci wewnętrznej/zewnętrznej, portów wejścia/wyjścia, plików, itd;
* komunikacyjne: zadania powinny mieć dostęp do komunikatów przesyłanych wewnątrz systemu;

Działanie RTAI

Ściśle ujmując, RTAI nie jest systemem czasu rzeczywistego, jak np. VXworks czy QNX, lecz “nakładką” do systemu Linux, bazującą na tym samym jądrze, pozwalającą na wykonywanie przez system zadań czasu rzeczywistego.

RTAI można wyobrazić sobie jako warstwę pomiędzy jądrem Linuksa a przychodzącymi do niego przerwaniami. Warstwa ta “przechwytuje” przerwania i przekierowuje je do jądra Linuksa. Przerwania sygnalizujące o potrzebie wykonania zadania czasu rzeczywistego dostają wyższy priorytet niż te sygnalizujące standardowe zadania Linuksa, przez co zadania czasu rzeczywistego zostają ustawione na czele kolejki zadań oczekujących na wykonanie, a dopiero po nich znajdują się zadania standardowe Linuksa.

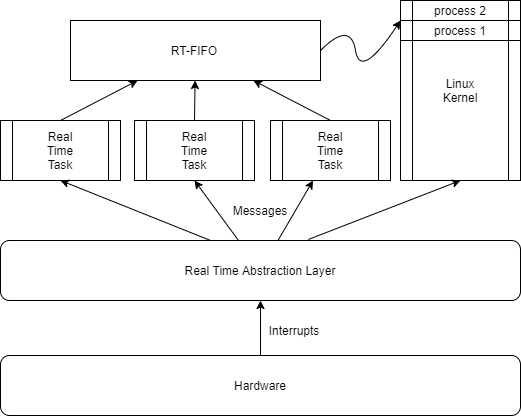
Zwykłe jądro działa tutaj jako zadanie tła. Wątek Linuksa nie może zablokować przerwań lub zapobiec byciu uprzedzonym przez wątek czasu rzeczywistego (RT). Jest to możliwe dzięki emulacji oprogramowania do sterowania przerwaniami.

Wątki czasu rzeczywistego są egzekwowane wewnątrz pamięci jądra Linuksa, i tym właśnie sposobem zapobiega to ich uprzedzeniu/zamianie przez zwykłe wątki jądra. Wątki RT są wykonywane w trybie nadzorczym procesora i mają pełen dostęp do *hardware’u*.

Jako, że RTOS (*Real-Time Operating System*) i aplikacje RTAI są “połączone” wspólną przestrzenią wykonawczą, mechanizm komunikacji systemowej jest zaimplementowany jako proste wywołanie funkcji, a nie jako przerwanie w pracy oprogramowania, przez co wydajność jest większa.

RTAI posługuje się kolejkami FIFO (“*First In - First Out*”). Kolejne zadania z kolejki są wykonywane na jądrze Linuksa, i dopiero, gdy kolejka procesów czasu rzeczywistego (RT) jest pusta, wykonywane są zwykłe zadania jądra.

Schemat RTAI jest przedstawiony poniżej. Widoczna jest nakładka “*Real Time Abstraction Layer*”, która, bazując na otrzymanych przerwaniach, rozpoczyna wątki RT i zwykłe wątki Linuksa.



*rys. 1. Schemat RTAI* (opracowanie własne)

Moduły RTAI

Koncept tego systemu opiera się na korzystaniu z ładowalnych modułów. Moduły te udostępniają całą funkcjonalność systemu.

Te moduły to:

* ***rtai*** - najważniejszy moduł systemu; jego zadaniem jest inicjalizacja wszystkich zmiennych oraz struktur danych, których system używa; ponadto wykonuje kopie adresów procedur obsługi przerwań systemowych, które zwykle są obsługiwane przez jądro Linux; ustawia konkretne procedury systemu jako procedury obsługi przerwań systemowych; aby go aktywować, należy go “zamontować” wywołaniem *rt\_mount\_rtai()*; należy również “zdemontować” moduł wywołując *rt\_umount\_rtai()*, gdy moduł nie jest już potrzebny; aktywacja tego modułu powoduje rozpoczęcie działania RTAI i od tego momentu przerwania sprzętowe są “filtrowane” przez RTAI, nawet jeśli nie używa się usług Linuksa; od momentu podłączenia modułu Linux nie zarządza ww. przerwaniami, funkcję tą przejmuje RTAI;
* ***rtai\_sched*** - w tym module zawarty jest tzw. “*scheduler*” systemu; po wywołaniu odpowiedniej funkcji systemowej przez mikroprocesor odbywa się przełączenie zadania; wybierane jest zadanie o najwyższym priorytecie, i zadanie to jest wykonywane; RTAI przyjmuje *0* za najwyższy priorytet, a *0x3fffffff* za najniższy; warto dodać, że zazwyczaj najniższy priorytet ma jądro Linux, gdyż działa ono jako zadanie tła (priorytet *0x7fffffff*); system RTAI wspiera tryb okresowej generacji przerwań oraz tryb generacji przerwań na żądanie (*“one-shot”*); dostępne są 3 typy “*schedulera*”:
* UP - tylko dla uniprocesorów;
* SMP - również dla multiprocesorów;
* MUP - tylko dla multiprocesorów;

Wszystkie funkcje modułu są dostępne niezależnie od rodzaju “*schedulera*”; przy załadowaniu tego modułu, automatycznie “montowany” jest moduł *rtai*;

* ***rtai\_fifos*** - zajmuje się implementacją kolejek FIFO; w tym module zaimplementowane są funkcje tworzenia, usuwania, czytania i pisania do kolejek FIFO interfejsu zadań czasu rzeczywistego; z punktu widzenia zwykłych procesów kolejki te są widoczne jako urządzenia znakowe wewnątrz katalogu */dev*; często moduł ten jest wykorzystywany pomimo tego, że nie jest konieczny, gdy korzysta się z innych modułów (np. *lxrt*); ma to miejsce, ponieważ kolejki FIFO są wygodnym narzędziem do komunikacji z systemem zarządzania przerwaniami - nie potrzebują do tego zamontowanego “*schedulera*”;
* ***lxrt*** - implementuje mechanizm pozwalający utrzymanie symetrii w komunikacji pomiędzy zadaniami czasu rzeczywistego i zwykłymi procesami Linux; dzięki niemu można wysyłać komunikaty: [RTAI - Linux; Linux - RTAI; RTAI - RTAI oraz Linux - Linux], używając do tego pamięci dzielonej;
* ***rtai\_shm*** - pozwala na dzielenie pamięci pomiędzy procesami czasu rzeczywistego i standardowymi procesami Linuksa; zapewnia ww. symetrię, tj. użycie tych samych wywołań zarówno przez procesy czasu rzeczywistego, jak i zwykłe procesy Linuksa; faktyczna alokacja ma miejsce przy pierwszej rezerwacji pamięci; każda kolejna alokacja to w rzeczywistości mapowanie większego obszaru pamięci w alokowanej już przestrzeni; analogicznie wygląda zwalnianie pamięci - obszar zarezerwowanej pamięci jest zmniejszany do momentu, w którym zwalnia się ostatni z obszarów, i wtedy następuje faktyczna dealokacja;
* ***rtai\_pthread*** - pozwala na zarządzanie procesami z twardymi ograniczeniami czasowymi; wszystkie te procesy są uruchomione w tej samej przestrzeni adresowej, a do tego mogą wykonywać pracę współbieżnie na dzielonych danych;
* ***rtai\_pqueue*** - zapewnia bezpieczne dostarczenie kolejek komunikatów;

Przerwania

Najważniejsze z operacji systemu RTAI dotyczą zarządzania przerwaniami. System przechwytuje wszystkie z przerwań. Jeżeli wystąpi przerwanie, którego procedura obsługi jest w standardowym jądrze Linuksa, wykonuje się tę procedurę, jeśli żaden proces czasu rzeczywistego nie jest wtedy aktywny. Jeżeli obsługa przerwania ma miejsce tylko w systemie czasu rzeczywistego, wykonuje się ją natychmiast. W przypadku, gdy obsługa przerwania znajduje się w systemie czasu rzeczywistego i w zwykłym jądrze Linuksa jednocześnie, to procedura czasu rzeczywistego jest wykonywana bezzwłocznie, a część, którą ma wykonać jądro Linuksa oznacza się tagiem “do wykonania”.

Odmierzanie czasu

W RTAI została dodana możliwość programowania mikrokontrolera (układ 8254), co pozwala na okresowe (periodyczne - “*periodic*”) generowanie przerwań oraz generowanie przerwań na żądanie (tzw. “*one shot*”). Kiedy korzystamy z trybu generowania przerwań na żądanie, czas mierzony jest na podstawie znacznika czasu procesora (*TSC*), a układ mikrokontrolera jest używany tylko do generacji przerwań.

Wspierane architektury

RTAI jest dostępne dla architektur:

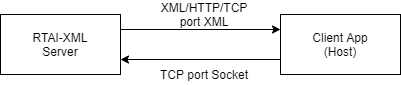
* *x86 (razem lub bez FPU i TSC);*
* *x86-64;*
* *Power PC;*
* *ARM (StrongARM, ARM7: clps711x-family, Cirrus Logic EP7xxx, CS89712, PXA25x);*
* *m68k (zarówno MMU, jak i NOMMU);*

RTAI-XML

Jest to składnik RTAI wdrażający zorientowany na usługi sposób projektowania i rozwoju aplikacji czasu rzeczywistego.

Działanie jest następujące: “*host”* łączy się z serwerem przez sieć TCP/IP przy pomocy protokołu bazującego na XMLu. Wchodzi w interakcję z “*targetem”*, na którym wykonywany jest proces czasu rzeczywistego. W ten sposób “*host”* może monitorować proces, śledzić generację sygnałów w systemie, przyciągnąć i zmienić parametry procesu, etc. Innymi słowy, pozwala na zdalną kontrolę aplikacji, będąc dodatkiem do RTAI i nie naruszając kluczowych funkcjonalności implementacji RT.

Schemat komunikacji RTAI-XML przedstawiony jest na zdjęciu poniżej:



*rys. 2. Schemat komunikacji RTAI-XML* (opracowanie własne)

Podobieństwo z RT-Linux

Warto przybliżyć podobieństwo RTAI do RT-Linux, który jest pierwszym pakietem *patchy* do jądra Linuksa i udostępniał możliwości systemu czasu rzeczywistego.

Oba systemy są oparte na wspólnej idei - zamiast modyfikować jądro Linuksa, wykorzystuje się do niego swojego rodzaju nakładkę, która pozwala uzyskać system czasu rzeczywistego.

RT-Linux nie jest dostępny dla architektury *m68k*.

Po wykupieniu przez firmę FSM Labs, rozwój RT-Linux został wstrzymany w maju 2001 roku, z ostatnim kompatybilnym jądrem Linux w wersji 2.4.4. RTAI jest wciąż rozwijanym rozwiązaniem, którego najnowsza wersja, RTAI 5.2, została wydana w maju 2019 roku.

Dodatkowo, RTAI jest rozwiązaniem “*open-source*”. RT-Linux był zaś produktem licencjonowanym.

RT-Linux nie pozwala na dynamiczną alokację pamięci. Pamięć musi zostać zarezerwowana przed rozpoczęciem wątku czasu rzeczywistego. RTAI natomiast dysponuje dynamicznym przydziałem przestrzeni w pamięci. Dodatkowo, w ramach pamięci dzielonej z jądrem Linuksa, RTAI dysponuje wspomnianym wcześniej modułem *rtai\_shm*, dzięki któremu można bezpiecznie dzielić pamięć z zapewnioną symetrią.

Źródła:

1. <https://www.rtai.org/>
2. <https://linuxdevices.org/ldfiles/misc/ripoll-rtl-v-rtai.html>