Rozdział 1

Obsługa przerwania zegarowego - algorytm do_timer i update_times

Rozdział 3

Autor:Jakub Jurkiewicz
e-mail:jj176743@zodiac.mimuw.edu.pl

W tym rozdziale zajmować się będziemy algorytmami funkcji i procedur związanych z obsługą przerwania zegarowego, oraz strukturami danych związanymi z tym przerwaniem.

1.1 Zmienne

Najpierw zajmijmy się zmiennymi wykorzystywanymi przez przerwanie zegarowe.

- jiffies - zmienna na której jest przechowywana ilość tyknięć zegara, które upłyneły od uruchomienia zegara.
- lost_ticks - ilość tyknięć zegara które upłyneły od ostatniego uruchomienia procedury update_times
- lost_ticks_system - j.w. ale w czasie gdy system nie był w trybie użytkownika
- xtime_lock - blokada przerwania zegarowego i wszystkich struktur z nim związanych
- TIMER_BH - znacznik przechowujący informację o konieczności wykonania dolnej połowy przerwania zegarowego
- TQUEUE_BH - znacznik informujący o konieczności wykonania dolnej połowę planisty

1.2 O wszystkim i o niczym - czyli ogólnie o przerwaniach i przerwaniu zegarowym

O tym jak kontroler przerwań reaguje na przerwanie powiedziane zostało w poprzednim rozdziale. Główną rzeczą na którą należy zwrócić uwagę jest idea dzielenia obsługi przerwania na dwie połowy. "Górna połowa" wykonuje rzeczy niezbędne, które muszą być wykonane natychmiast. Musi ona działać jak najszybciej gdyż w trakcie jej wykonywania pozostałe przerwania są zablokowane (uwaga: przerwanie zegarowe ma w Linuksie najwyższy priorytet). Jeśli zachodzi konieczność wywołania dolnej połowy zaznacza ona to na odpowiednich zmiennych. Do tego służą zmienne TIMER_BH i TQUEUE_BH. "Połowa dolna" natomiast może działać wolniej, a w jej trakcie obsługa przerwań jest z założenia włączona (oczywiście może ona wyłączać przerwania). W Linuksie jest nią timer_bh oraz tqueue_bh. Druga z nich to obsługa kolejki zadań czym nie będę się w tym rozdziale zajmował. Pierwsza z nich natomiast wywołuje update_times i obsługę czasomierzy.

1.3 Górna połowa

Większość rzeczy napisanych w tej sekcji (poza opisem procedury do_timer) jest specyficzne dla architektury komputera, ja zaś będę się zajmował architekturą procesora i386.

Po tym krótkim wstępie zajmijmy się przerwaniem zegarowym i jego losami. Żeby mówić o przerwaniu urządzenie musi wysłać jakiś sygnał do kotrolera przerwań informując o jakimś zdarzeniu. Tym zderzeniem będzie drgnięcie zegara. Kontroler przerwań wykonuje szereg działań (bliższe szczegóły opisane w poprzednim rozdziale) i wywołuje procedurę timer_interrupt.

1.3.1 Algorytm timer_interrupt

Procedura ta blokuje xtime_lock. Ewentualnie kotroluje cykle procesora pentium i woła do_timer_interrupt. A na koniec odblokowuje xtime_lock.

1.3.2 Algorytm do_timer_interrupt

Procedura ta woła procedurę do_timer. Następnie synchronizuje zegary, odpowiednio poprawiwjąc czas. Robi też ona wiele rzeczy specyficznych dla różnych architektur. (włączone/wyłączone SMP itd.)

1.3.3 Algorytm do_timer

Procedura ta jedyna z obsługiwanych w górnej połowie przerwania jest niezależna od sprzętu. Działa ona szybko, ale także robi niewiele.

1.3.4 Końcówka wspólna dla wszystkich przerwań

Na koniec, kiedy procedura timer_interrupt zakończy działanie odbywa się skok pod etykietę ret_from_intr (zawartą w pliku entry.S) gdzie wywoływane są "dolne połowy" i ewentualnie planista (schedule). W procedurze tej odblokowywane są przerwania tak, że dolne połowy, oraz

Algorytm 1 do_timer

Dane wejściowe: struct pt_regs -stan rejestrów.
Wynik: jest typu void nie zwraca nic jedynie modyfikuje stan komputera.

```
zwiększ jiffies o jeden
zwiększ lost_ticks o jeden
zaznacz konieczność wykonania dolnej “połowy”
jeśli Nie jesteś w trybie użytkownika (poznasz to po rejestrach) to
   zwiększ lost_ticks_system o jeden
koniec jeśli
jeśli Zarejestrowano zadanie dla planisty to
   zaznacz konieczność wykonania “dolnej połowy” planisty
koniec jeśli
```

planista są wykonywane z właczonymi przerwanami. (Szczegółowe informacje o ret_from_intr można znaleźć w poprzednim rozdziale, a o planiście w rozdziałach 10, 11 i 12).

1.4 Dolna połowa - timer_bh

Tu opiszemy tylko główną procedurą. Jest ona krótka w treści, ale procedury przez nią wołane robią dużo. Jest ona wywokonywana, gdy system dowie się, że zostało zaznaczone TIMER_BH, najczęściej w trakcie końca obsługi przerwania zegarowego, gdy wszystkie przerwania są już włączone.

Algorytm 2 timer_bh

Dane wejściowe: NIC
Wynik: NIC - typu void

```
uruchom funkcję update_times
uruchom stare czasomierze
uruchom listę czasomierze
```

1.4.1 To tu najwięcej się robi - Algorytm update_times

Procedura działa w czasie gdy obsługa przerwań jest włączona. Nie musi się więc zajmować stanem rejestrów i flag. Wszystkie rzeczy natomiast, poza odświezaniem czasu procesu, wykonuje z wyłączonym przerwaniem zegarowym.

1.4.2 O najbardziej ukrytych

Tu zostaną przedstawione procedury wołane przez upadate_times.

calc_load

Procedura ta służy liczeniu odpowiednich statystyk aktywnych zadań i ich czasu działania.

Algorytm 3 update_times

Dane wejściowe: NIC
Wynik: NIC - jak zwykle

wprowadz zmienną ticks;
Zablokuj obsługę przerwania zegarowego
NA ticks przypisz lost_ticks
wyzeruj lost_ticks
jeśli ilosć tyknięć większa od zera - ticks >0 **to**
    wprowadź zmienną system
    przypisz na nią lost_ticks_system, a na lost_ticks_system zero
    Zmodyfikuj stosowne statystyki przy uzyciu zmiennej ticks.
    Wykonaj update_wall_time
    Odblokuj przerwanie zegarowe
    uaktualnij czasy procesu (wywołaj update_process_times)
w przeciwnym przypadku
    Odblokuj przerwanie zegarowe
koniec jeśli

update_process_times

Procedura ta zajmuje się uaktualnieniem czasu procesu, oraz sprawdzeniem czy nie przekroczył on na tyle limitów, że powinniśmy wysłać mu sygnału (reakcją na który najczęściej jest zakonczenie procesu).

update_wall_time

Procedura ta dokonuje poprawek czasu i zmiennych z nim związanych, oraz sprawdza czy nie zostały przekroczony zakresy zmiennych obsługujących czas.

1.5 Zadania

1. Dlaczego obsługę przerwania zegarowego podzielono na dwie części?
2. Jakie przerwania są włączone w czasie wykonania update_times?
3. Jak jest związana procedura update_times z do_timer?
4. Kiedy proces jest wywłaszany z procesora na skutek przekroczenia czasu?

1.6 Rozwiązania zadań

1. Ze względu na to, że część wykonywana z wyłączonymi wszystkimi przerwaniami musi być szybko wykonywana.
2. Włączone są wszystkie przerwania, z wyjątkiem przerwania zegarowego które jest wyłączane na pierwszą część działania tej funkcji.

3. W procedurze do_timer zaznaczana jest konieczność wykonania funkcji timer_bh która wywołuje update_times.

4. W procedurze schedule wywoływanej w trakcie powrotu z przerwania.

Literatura

[Rochkind 93] Rochkind M. J.: *Programowanie w systemie Unix dla zaawansowanych.* Warszawa, WNT 1993.

[Stevens 95] Stevens W. R.: *Programowanie zastosowań sieciowych w systemie Unix.* Warszawa, WNT 1995.

[Rubini 99] Rubini A.: *Linux sterowniki urządzeń.* Warszawa, RM 1999.

[] Kod źródłowy Linuxa Kernel 2.2.17: *pliki kernal/sched.c oraz z katalogu arch/i386/kernel irq.h time.c entry.S*