Zastosowanie `clock_*`, `timer_*`

1. Wstęp

Celem zadania jest stworzenie programu, korzystającego z wielu „budzików" równocześnie, z których część działa cyklicznie. Program ten ma być tak skonstruowany, by minimalizować opóźnienia w obsłudze „dzwonków". Aby uzyskać potrzebne informacje diagnostyczne, będzie on gromadził informacje o dokładności swojego działania.

2. Specyfikacja

2.1. Argumenty

Program przyjmuje następujące argumenty:
— `--total <float>` - obowiązkowy,
— `--update <float>`- obowiązkowy,
— n liczb `<float>`, gdzie $n > 0$.
Każdy z parametrów określa czas w decysekundach.

2.2. Działanie

Działanie programu składa się z trzech faz: przygotowań, wykorzystania budzików, podsumowań.

Faza środkowa.
1. Ta część ma trwać dokładnie tyle, ile zostało podane w parametrze `--total`.
2. Każdemu z (nienazwanych) parametrów liczbowych odpowiada jeden licznik całkowitoliczbowy. Licznik ten, w ustalonych przez parametr odstępach czasu, jest zwiększany o 1 (wartość początkowa 0).
   Dodatkowo odnotowywane są następujące informacje:
   — sumaryczna i maksymalna (tj. osobne wartości) ilości zaległych zmian (`timer_getoverrun`),
   — kumulacja różnic i maksymalna różnica między oczekiwanymi a faktycznymi momentami obsługi dzwonka.
3. Cyklicznie, w odstępach podanych w parametrze `--update`, jest wysyłany na standardowe wyjście łańcuch z opisem zgromadzonych wyników.
   Obsługa tej operacji nie może być przerwana żadnym, generowanym przez ten proces, sygnałem.

Faza końcowa, oprócz właściwych dla niej czynności, wyświetla informację o różnicy między oczekiwanym, a (w miarę możliwie dokładnie wyznaczonym) rzeczywistym czasem wykonania fazy środkowej.

3. Uzupełnienie

Należy przygotować zestawy parametrów ilustrujące różne przypadki opóźnień.