Programowanie równoległe.

Przetwarzanie równoległe i rozproszone. Laboratorium 3

Wykonał: Mikita Shmialiou

Cel:

* nabycie praktycznej umiejętności manipulowania wątkami Pthreads – tworzenia, niszczenia, elementarnej synchronizacji
* przetestowanie mechanizmu przesyłania argumentów do wątku
* poznanie funkcjonowania obiektów określających atrybuty wątków.

Wykonanie ćwiczenia

1. Pobranie pliku „pthreads\_detach\_kill.c”

A screenshot of a computer

AI-generated content may be incorrect.

1. Uzupełnienie kodu programu   
   A screen shot of a computer program

   AI-generated content may be incorrect.
2. Testowanie programu  
   A screenshot of a computer

   AI-generated content may be incorrect.
3. Pytania:
4. W jakich dwóch trybach mogą funkcjonować wątki Pthreads?

* Joinable (łączalne / dołączalne)
* Detached (odłączone)

1. Jaka jest różnica między tymi trybami?

Odłączone wątki działają niezależnie od innych wątków i nie mogą czekać na inne wątki. Czekanie na inne wątki jest realizowane za pomocą funkcji pthread\_join() (więcej później). Wątki dołączalne mogą wywołać pthread\_join(), aby czekać na inny wątek.

1. Kiedy wątek standardowo kończy swoje działanie?

Wątek standardowo kończy swoje działanie po zakończeniu procedury (funkcji) punktu wejścia, czyli po wykonaniu wszystkich instrukcji, które zostały dla niego zdefiniowane.

1. W jaki sposób można wymusić zakończenie działania wątku? (czym różnią się w tym przypadku wątki odłączone i standardowe?)

Wątek może zakończyć działanie w dowolnym momencie, wywołując podprogram pthread\_exit. Wywołanie podprogramu exit kończy cały proces, włączając wszystkie wątki.

Joinable: Można użyć pthread\_join po anulowaniu, żeby potwierdzić zakończenie i odzyskać kod.

Detached: Nie da się sprawdzić rezultatu anulowania — po zakończeniu wątek sam czyści zasoby.

1. Jak wątek może chronić się przed próbą "zabicia"? Jak można sprawdzić czy próba "zabicia" wątku powiodła się? (czym różnią się w tym przypadku wątki odłączone i standardowe?)

Najprostszy i główny sposób, w jaki wątek może chronić się przed anulowaniem (próbą "zabicia"), to tymczasowe wyłączenie możliwości anulowania za pomocą funkcji:

pthread\_setcancelstate(PTHREAD\_CANCEL\_DISABLE,NULL)

Wątek wywołuje tę funkcję przed wejściem do krytycznego fragmentu kodu, a po jego zakończeniu przywraca możliwość anulowania przez:

pthread\_setcancelstate(PTHREAD\_CANCEL\_ENABLE, NULL)

Gdy anulowanie jest wyłączone, żądania pthread\_cancel są wstrzymywane i nie przerywają działania wątku.

1. Zaprojektowanie i utworzenie nowej procedury tworzenia wątków

Kod:   
A screen shot of a computer program

AI-generated content may be incorrect.

Testowanie:

A screen shot of a computer

AI-generated content may be incorrect.

Jak poprawnie przekazać identyfikator do wątku?

Poprawne przekazywanie odbywa się przez przekazanie wskaźnika do oddzielnej zmiennej dla każdego wątku – najczęściej tablicy int identyfikatory[N], gdzie &identyfikatory[i] jest unikalne dla każdego wątku.

Sprawozdanie: Równoległe obliczanie całki metodą trapezów

Wykonał: Mikita Shmialiou

1. Cel ćwiczenia

Celem laboratorium było zaimplementowanie programu obliczającego całkę oznaczoną funkcji sinus w przedziale [0, π] metodą trapezów, wykorzystując trzy różne podejścia:

* Wersja sekwencyjna - obliczenia na jednym wątku
* Zrównoleglenie pętli - podział iteracji między wątki
* Dekompozycja obszaru - podział przedziału całkowania między wątki

2. Metodyka

2.1. Algorytm całkowania

Wykorzystano metodę trapezów:

double calka = 0.0;

for(int i=0; i<N; i++) {

double x1 = a + i\*dx;

calka += 0.5\*dx\*(funkcja(x1)+funkcja(x1+dx));

}

2.2. Implementacja równoległa

Zrównoleglenie pętli:

* Każdy wątek oblicza sumę cząstkową dla przypisanych iteracji
* Wyniki sumowane są z użyciem mutexa

Dekompozycja obszaru:

* Każdy wątek oblicza całkę w swoim podprzedziale
* Wyniki sumowane są bez konieczności synchronizacji

3. Wyniki

3.1. Testy dla dx = 0.1

|  |  |  |
| --- | --- | --- |
| Metoda | Czas wykonania [s] | Wynik |
| Sekwencyjna | 0.000033 | 1.998393360970144 |
| Zrównoleglenie pętli (4w) | 0.000292 | 1.998393360970145 |
| Dekompozycja obszaru (4w) | 0.000157 | 1.998393360970145 |

3.2. Testy dla dx = 0.000001

|  |  |  |
| --- | --- | --- |
| Metoda | Czas wykonania [s] | Wynik |
| Sekwencyjna | 0.029032 | 1.999999999999758 |
| Zrównoleglenie pętli (4w) | 0.008087 | 1.999999999999837 |
| Dekompozycja obszaru (4w) | 0.007473 | 1.999999999999847 |

4. Analiza

4.1. Poprawność wyników

Wszystkie metody zwracają identyczne wyniki, co potwierdza poprawność implementacji. Dla dx = 0.000001 wynik jest bardzo bliski teoretycznej wartości 2.0.

4.2. Wydajność

Przyspieszenie: Dla 4 wątków osiągnięto przyspieszenie ~3.5x

Różnice między metodami:

* Dekompozycja obszaru jest nieco szybsza od zrównoleglenia pętli
* Różnica wynika z braku konieczności synchronizacji w metodzie dekompozycji

4.3. Wnioski

* Dla dużych N (małe dx) równoległość daje znaczące przyspieszenie
* Dekompozycja obszaru jest bardziej efektywna dla całkowania numerycznego
* Im większa liczba trapezów, tym większy zysk z równoległości