Programowanie równoległe. Przetwarzanie równoległe i rozproszone.
Laboratorium 3

Cel:

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

Kroki:

1. Utworzenie katalogu roboczego (np. *lab_3*)
2. Pobranie pliku „*pthreads_detach_kill.c*" ze strony WWW, rozpakowanie (w odrębnym podkatalogu np. *zad_1*)
3. Uzupełnienie kodu programu *pthreads_detach_kill.c* we wskazanych miejscach, zgodnie z opisem w p. 2.5.3 skryptu znajdującego się na stronie WWW (**ze zrozumieniem**, w przypadku niejasności skonsultowanie z odpowiednimi fragmentami skryptu lub prowadzącym)
4. Przetestowanie działania kodu **(ocena)**

- *-> W jakich dwóch trybach mogą funkcjonować wątki Pthreads? Jaka jest różnica między tymi trybami? Kiedy wątek standardowo kończy swoje działanie? W jaki sposób można wymusić zakończenie działania wątku? (czym różnią się w tym przypadku wątki odłączone i standardowe?) 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?)*

5. Zaprojektowanie i utworzenie (w odrębnym podkatalogu np. *zad_2*) nowej (nowej tzn. napisanej „od zera", bez elementów z poprzednich punktów) procedury wątków, do której jako argument przesyłany jest identyfikator każdego wątku, z zakresu *0..liczba_wątków-1*, wątek powinien wypisywać na ekranie swój systemowy ID (procedura *pthread_self()*) oraz swój przesłany identyfikator
6. Przetestowanie poprawnego działania programu (w tym poprawnego przesyłania indywidualnych identyfikatorów) dla różnej liczby tworzonych i **współbieżnie** działających wątków (należy tworzyć wątki w pętli, a następnie, w kolejnej pętli, oczekiwać na ich zakończenie) – **wnioski dotyczące warunków poprawności przesyłania identyfikatorów do wątków powinny znaleźć się w sprawozdaniu (ocena)**

- *-> W jaki sposób można poprawnie przesłać identyfikator do wątku? Jaki może pojawić się błąd synchronizacji w przypadku próby przesłania zwykłego wskaźnika do liczby całkowitej?*
- *W celu odpowiedzi na pytanie powyżej najlepiej w kodzie dokonać modyfikacji, tak żeby przesyłać do wątków adres zmiennej będącej zmienną sterującą pętli tworzenia wątków i uzyskać w trakcie wykonania błąd: co najmniej dwa wątki wypisujące ten sam identyfikator (uzyskanie błędu może wymagać wielokrotnego uruchomienia programu z różnymi liczbami tworzonych wątków)*
- *-> W sprawozdaniu należy umieścić wydruki terminala (np. zrzuty ekranu z identyfikacją użytkownika terminala): jeden z programu poprawnie przesyłającego identyfikatory do wątków i drugiego, w którym pojawia się błąd synchronizacji*

---------- *3.0* ----------

Dalsze kroki dla podniesienia oceny:

1. Zaprojektowanie i utworzenie (w odrębnym podkatalogu np. *zad_3*) nowego programu z procedurą wątków, do której jako argument przesyłany byłby wskaźnik do samodzielnie zaprojektowanej struktury zawierającej co najmniej 2 pola, pole oznaczone jako wejściowe (*in*) i pole oznaczone jako *in/out*
    - najlepszą formą realizacji tego punktu jest przygotowanie szkieletu do realizacji zadania z lab 5 – obliczanie całki w przedziale (a,b) z zadanej funkcji
        - każdy wątek jako argument dostaje wskaźnik do swojej struktury z danymi wejściowymi (funkcja tworząca wątki musi stworzyć tablicę struktur z argumentami dla każdego wątku)

- każdy wątek pracuje nad sobie przydzielonym fragmentem przedziału (a,b) – funkcja tworząca wątki musi podzielić przedział na tyle fragmentów ile jest wątków
- pojedyncza struktura zawiera współrzędną początku fragmentu, współrzędną końca fragmentu oraz pole na wynik (wszystkie typu double)
- każdy wątek wykonuje pewną operację (np. oblicza średnią z przesłanych współrzędnych, czyli współrzędną środka fragmentu odcinka), a wynik zapisuje w odpowiednie pole struktury
- funkcja tworząca wątki oczekuje na zakończenie pracy wątków, a następnie sumuje wartości z pola wynikowego struktur każdego z wątków (i wypisuje na ekranie)

2. Przetestowanie działania kodu dla co najmniej 2 wątków

---------- 4.0 ----------

Dalsze kroki dla podniesienia oceny:

1. Na podstawie dotychczasowych programów stworzenie nowego kodu (w odrębnym podkatalogu np. *zad_4*), w którym uruchomione zostanie kilka wątków z różnymi zadanymi wartościami atrybutów, a następnie parametry te zostaną wypisane w procedurze wątku (procedura „*pthread_getattr_np*” lub podobne). Przetestowanie działania atrybutów takich jak np.:
   - rozmiar i położenie stosu (przetestowanie działania: statyczna alokacja dużej tablicy w procedurze wątku (np. int tab[10000000] = {0}; - należy kompilować z opcją *-g* , żeby uniknąć optymalizacji) i sprawdzenie jaki trzeba dobrać rozmiar stosu, aby procedura mogła zostać zrealizowana) **(ocena)**
   - priorytet wykonania
   - afiniczność – możliwość sterowania wyborem rdzenia, na którym zostanie uruchomiony wątek
   - inne (dostępne procedury można znaleźć posługując się np. stroną podręcznika „*man pthread_attr_init*” i innymi wymienionymi tam stronami) **(ocena)**
     - *-> jakie atrybuty tworzonych wątków są możliwe do określenia w wersji biblioteki Pthreads zainstalowanej w systemie? Jakie jest znaczenie poszczególnych atrybutów?*

   Dowolne zadania polecane przez prowadzących

---------- 5.0 ----------

Warunki zaliczenia:

1. Obecność na zajęciach i wykonanie kroków 1-6.
2. Oddanie krótkiego sprawozdania o treści i formie zgodnej z regulaminem ćwiczeń laboratoryjnych (z opisem zadania, utworzonymi przez siebie uzupełnieniami kodów źródłowych programów i przykładowymi wydrukami pokazującymi wynik działania programów – wklejone jako obrazy z identyfikacją osoby przeprowadzającej obliczenia – zgodnie z regulaminem laboratoriów).
3. Symbol -> oznacza pytania, na które odpowiedzi ma dać laboratorium (odpowiedzi powinny znaleźć się w sprawozdaniu)