Temat zajęć	Wątki w systemie Linux
Zakres materiału	Biblioteka *pthreads* i jej wykorzystanie do tworzenia programów wielowątkowych

Materiał teoretyczny

- wątki i muteksy w Linuksie
- typ *pthread_t,* funkcje *pthread_create(), pthread_join(), pthread_self()*
- funkcje *mutex_lock() i mutex_unlock()*
- pomiar czasu, funkcja *clock_gettime()*

Treść zadania

Napisać języku C program spełniający poniższe wymagania:

- program akceptuje dokładnie dwa argumenty wywołania; oba są liczbami całkowitymi i oznaczają odpowiednio: pierwszy – liczbę ***n*** danych rzeczywistych przetwarzanych przez kod i liczbę ***w*** wątków, w których odbędzie się przetwarzanie; zakładamy, że: $1 < n < 10000000$ oraz $1 < w < n$;
- program tworzy tablicę danych typu *float* o rozmiarze ***n*** i wypełnia ją danymi według poniższego schematu:

```
srand(0);
for(int i = 0; i < n; i++)
   t[i] = 1000. * rand() / RAND_MAX;
```

- program powołuje do życia ***w*** wątków i przekazuje każdemu z nich inny fragment tablicy ***t***;
- jeżeli ***n*** nie dzieli się bez reszty przez ***w***, to ***n*-1** pierwszych wątków otrzymuje równe co do rozmiaru fragmenty tablicy, a wątek ostatni dostaje fragment powiększony o resztę z tego dzielenia;
- każdy wątek rozpoczynając pracę wypisuje na *stdout* swój identyfikator i liczbę elementów w swoim fragmencie tablicy;
- każdy wątek oblicza **sumę** elementów w swoim fragmencie tablicy, a następnie aktualizuje wspólną, globalną zmienną, która w efekcie będzie sumę elementów całej tablicy (**uwaga!** aktualizacja tej zmiennej jest **sekcją krytyczną** i musi być chroniona muteksem);
- każdy z wątków na zakończenie pracy wyprowadza na *stdout* swój identyfikator i obliczoną sumę częściową;
- program czeka na zakończenie wszystkich wątków, a następnie wypisuje wartość wspólnej zmiennej globalnej oraz czas jaki upłynął od włączenia pierwszego wątku do zakończenia ostatniego; pomiar czasu w każdym przypadku musi zostać wykonany przy użyciu funkcji *clock_gettime()*;
- następnie program jeszcze raz sumuje dane w tablicy, tym razem bez użycia wątków, i na zakończenie wyświetla ponownie przeliczoną sumę i czas, jaki zajęło ponowne sumowanie.

- Przykładowe wyjście z programu:

```
$ ./prog 1000 3
Thread #140613932529408 size=333
Thread #140613924136704 size=333
Thread #140613932529408 sum=174920.453125
Thread #140613924136704 sum=169507.859375
Thread #140613806716672 size=334
Thread #140613806716672 sum=163697.187500
w/Threads: sum=508125.500000, time=3.23sec
wo/Threads: sum=508125.500000, time=6.898sec
```

Uwaga! Kod źródłowy programu (1 plik) musi zostać jako **załącznik** przesłany na adres `son1@wi.zut.edu.pl`:

- plik z kodem źródłowym musi mieć nazwę: `numer_indeksu.so.lab07.c` (np. `66666.so.lab07.c`),
- plik musi zostać wysłany z poczty uczelnianej (domena `zut.edu.pl`),
- temat maila musi mieć postać:
  `SO IN1 99X LAB07`
  gdzie *99X* to numer grupy laboratoryjnej (np. `SO IN1 20A LAB07`),
- w pierwszych trzech liniach kodu źródłowego w komentarzach (każda linia komentowana osobno) musi znaleźć się:
  - informacja identyczna z zamieszczoną w temacie maila,
  - imię i nazwisko osoby wysyłającej maila,
  - adres e-mail, z którego wysłano wiadomość
    np.:

```
// SO IN1 20A LAB07
// Jan Nowak
// nj66666@zut.edu.pl
```

- e-mail nie może zawierać żadnej treści (tylko załącznik).

Dostarczone kody programów będą analizowane pod kątem wykrywania plagiatów. Niewysłanie wiadomości, wysłanie jej w formie niezgodnej z powyższymi wymaganiami lub wysłanie pliku, który nie będzie się kompilował i uruchamiał, będzie traktowane jako brak programu i skutkowało otrzymaniem oceny niedostatecznej.