INSTYTUT TELEINFORMATYKI I AUTOMATYKI

Wydział Cybernetyki WAT

Przedmiot: SYSTEMY OPERACYJNE

SPRAWOZDANIE Z ĆWICZENIA LABORATORYJNEGO Nr 6

Temat ćwiczenia**: WĄTKI W SYSTEMACH LINUX/UNIX**

Wykonał:

Piotr Matyjek

Grupa: **I6Y3S1**

Ćwiczenie wykonane dnia
10.11.2017

Prowadzący ćwiczenie
mgr. inż. Stanisław Matusiak

Ocena:
……

Opis rozwiązania

Przydzielony został mi numer zadania 1. Kod programu realizującego zadanie został zamieszczony poniżej. W celu przesłania do procedury, którą będzie wykonywał wątek, zarówno wartości „x" jak i również wartości „i", utworzyłem strukturę danych „ele", którą od razu przypisałem nazwę alternatywną „elementy" przy użyciu typedef, aby móc później w programie odnosić się do struktury po krótszej nazwie. Procedura którą wątek wykonuje przyjmuje jeden pusty wskaźnik, który następnie jest rzutowany na wskaźnik do struktury wspomnianej wcześniej. Dalej w procedurze są wykonane odpowiednie deklaracje i przypisania wartości zmiennych i alokowanie pamięci dla „wynik", aby po zakończeniu wykonywania wątku, nie została zwolniona ta część pamięci. To pozwoli przesłać „wynik" przy użyciu pthread_exit do wątku głównego, który przy użyciu pthread_join będzie mógł odczytać przesłaną wartość.

Program został tak zbudowany, żeby wartości „x" i „N" były podawane jako argumenty dla uruchomienia programu w konsoli, czyli: „./matyjek_threads.o x N".
Konwersja podanych argumentów z typów znakowych na typy double i int odpowiednio dla „x" i „N", została wykonana za pomocą funkcji sscanf (zostało to opatrzone odpowiednim komentarzem w kodzie programu). Ponieważ nie wiadomo jakie „N" użytkownik wpisze, uznałem, że najkorzystniejszym rozwiązaniem dla utworzenia odpowiedniej ilości identyfikatorów wątków i odpowiedniej ilości struktur „elementy", będzie utworzenie tablic dynamicznych, przy użyciu funkcji malloc.
Wypełnianie pojedynczych obiektów struktury „elementy" jest realizowane w tej samej pętli co tworzenie wątków, w których wykorzystywany jest ten obiekt struktury.
W kolejnej pętli wykonywany jest pthread_join, do którego przekazywany jest „wynik" z wątku, a następnie dodawany do „sumy". W międzyczasie wyświetlany jest identyfikator wątku kończącego pracę. Ostatnią instrukcją w pętli jest free(wynik). Została ona zaimplementowana, w celu uniknięcia wycieków pamięci. Tak samo free(L) i free(T) zostały zaimplementowane, aby zwolnić pamięć po ich wykorzystaniu. Następnie na standardowe wyjście wyświetlana jest suma szeregu obliczonego w programie, a w kolejnej linijce wyświetlony zostaje wynik funkcji cosinus z „x", w celu weryfikacji poprawności.

Kod źródłowy programu

```c
#include <stdio.h>
#include <pthread.h>
#include <math.h>
#include <stdlib.h>
#include <unistd.h>

typedef struct ele
{
double x;
int i;
}elementy;



void *wyraz (void *arg)
{
elementy *point=(elementy*)arg;

int i, f,g;
double iks_kw, minus_jeden, *wynik, x, silnia=1;

i=point->i;
x=point->x;

wynik=(double*)malloc(sizeof(double));

f=2*i;

//printf("%f  ",point->x);
minus_jeden=pow((double)(-1),(double)i);
iks_kw=pow(x,(double)(2*i));                              //zmiana1

for(g=1;g<=f;g++)
{
silnia=silnia*g;
}

*wynik=(minus_jeden*iks_kw)/silnia;
//printf("%f   ", *wynik);                                zmiana2

pthread_exit(wynik);



}
```

```c
int main(int argc, char *argv[])
{
int N,  i;
double *wynik, suma=0, x;
elementy *T;
pthread_t *L;

sscanf(argv[1],"%lf",&x);                       //konwersja pierwszego argumentu na double
sscanf(argv[2],"%d",&N);                        //konwersja drugiego argumentu na int

T=malloc(sizeof(elementy)*(N+1));
L=malloc(sizeof(pthread_t)*(N+1));

for(i=0;i<=N;i++)
{
T[i].x=x;
T[i].i=i;
//printf("%f   ", x);
pthread_create(&L[i],NULL, wyraz,&T[i]);

}
sleep(2);

for(i=0;i<=N;i++)
{
pthread_join(L[i], (void**)&wynik);

printf("%ld\n", (long)L[i]);

suma=suma+(*wynik);

free(wynik);                            //zwalnianie pamieci dla wynik

}

free(T);                                //zwalnianie pamięci dla tablicy struktur
free(L);                                //zwalnianie pamięci dla tablicy wątków
printf("wynik: %f\n",suma);

printf("%f\n", cos(x));                 //wypisanie wartosci cos(x) dla sprawdzenia poprawności

return 0;
}
```

Wyniki uruchomienia

```
piotr@Piotr-Debian:~/Dokumenty$ gcc -lpthread matyjek_threads.c -o matyjek.o -lm
piotr@Piotr-Debian:~/Dokumenty$ ./matyjek.o 3.14 10
-1219888320
-1228281024
-1236673728
-1245066432
-1253459136
-1261851840
-1270244544
-1278637248
-1287029952
-1295422656
-1303815360
wynik: -0.999999
-0.999999
piotr@Piotr-Debian:~/Dokumenty$
```

```
piotr@Piotr-Debian:~/Dokumenty$ ./matyjek.o 6.28 10
-1219204288
-1227596992
-1235989696
-1244382400
-1252775104
-1261167808
-1269560512
-1277953216
-1286345920
-1294738624
-1303131328
wynik: 1.000293
0.999995
piotr@Piotr-Debian:~/Dokumenty$
```

```
piotr@Piotr-Debian:~/Dokumenty$ ./matyjek.o 1.57 8
-1219949760
-1228342464
-1236735168
-1245127872
-1253520576
-1261913280
-1270305984
-1278698688
-1287091392
wynik: 0.000796
0.000796
piotr@Piotr-Debian:~/Dokumenty$
```

```
piotr@Piotr-Debian:~/Dokumenty$ ./matyjek.o 1.57 99
-1219908800
-1228301504
-1236694208
-1245086912
-1253479616
-1261872320
-1270265024
-1278657728
-1287050432
-1295443136
-1303835840
-1312228544
-1320621248
-1329013952
-1337406656
-1345799360
-1354192064
-1362584768
-1370977472
-1379370176
-1387762880
-1396155584
-1404548288
-1412940992
-1421333696
-1429726400
-1438119104
-1446511808
-1454904512
-1463297216
-1471689920
-1480082624
-1488475328
-1496868032
-1505260736
-1513653440
-1522046144
-1530438848
-1538831552
-1547224256
-1555616960
-1564009664
-1572402368
-1580795072
-1589187776
-1597580480
```

```
-1605973184
-1614365888
-1622758592
-1631151296
-1639544000
-1647936704
-1656329408
-1668285632
-1676678336
-1685071040
-1693463744
-1701856448
-1710249152
-1718641856
-1727034560
-1735427264
-1743819968
-1752212672
-1760605376
-1768998080
-1777390784
-1785783488
-1794176192
-1802568896
-1810961600
-1819354304
-1827747008
-1836139712
-1844532416
-1852925120
-1861317824
-1869710528
-1878103232
-1886495936
-1894888640
-1903281344
-1911674048
-1920066752
-1928459456
-1936852160
-1945244864
-1953637568
-1962030272
-1970422976
-1978815680
-1987208384
-1995601088
-2003993792
-2012386496
-2020779200
-2029171904
-2037564608
-2045957312
-2054350016
wynik: 0.000796
0.000796
piotr@Piotr-Debian:~/Dokumenty$
```

PODSUMOWANIE

Program został napisany w języku C i uruchomiony w środowisku systemu Linux Debian ver. 9. Program wykonuje się poprawnie, jak widać na zamieszczonych powyżej przykładach wywołania programu. Dla większych wartości „x” wynik działania programu będzie się trochę różnił od wyniku funkcji cos(x), ponieważ występują błędy zaokrągleń. Dla większych wartości „N” występują błędy, które uniemożliwią poprawne obliczenie wartości szeregu i wykonania programu. Takie błędy napotkałem, gdy próbowałem podać argument „N”, który był większy od 370. W kodzie programu

powyżej zostały dokonane niewielkie zmiany w stosunku do pliku, który wysłałem Panu w dniu zajęć. Są to:
- dodanie trzech funkcji free, opisanych komentarzem w kodzie programu, jak i również w „Opisie Rozwiązania”;
- usunięcie rzutowania zmiennej x na typ double w procedurze „wyraz” (komentarz „zmiana1”);
- zakomentowanie funkcji printf w procedurze „wyraz”. Wyświetlała ona wyniki pojedynczych wyrazów szeregu, w celu weryfikacji poprawności wykonywania programu;
- rozdzielanie fragmentów kodu w celu zwiększenia czytelności.

Wysyłam Panu wraz ze sprawozdaniem kod programu wraz z powyższymi zmianami.