5. Czekanie na zakończenie wątku

- Przejrzyj, skompiluj (użyj flagi -lm) i uruchom program join.c.
- Zmodyfikuj program tak, aby wątki zwracały różne wartości. Skompiluj i uruchom. Działa? Dla porównania, przejrzyj, skompiluj i uruchom program przykładowy detached.c.
- Obserwuj zachowanie i zauważ, że w przykładzie nie ma instrukcji join.
- Wyjaśnij różnicę między tymi dwoma programami.

```
adrian@Adrian:~/Desktop/SCR/SCR_77/kod1$ ./join.out
Main: creating thread 0
Main: creating thread 1
Main: creating thread 2
Thread 0 starting...
Main: creating thread 3
Thread 2 starting...
Thread 3 starting...
Thread 1 starting...
Thread 0 done. Result = -3.153838e+06
Main: completed join with thread 0 having a status of 0
Thread 2 done. Result = -3.153836e+06
Thread 3 done. Result = -3.153835e+06
Thread 1 done. Result = -3.153837e+06
Main: completed join with thread 1 having a status of 1
Main: completed join with thread 2 having a status of 2
Main: completed join with thread 3 having a status of 3
Main: program completed. Exiting.
```

```
adrian@Adrian:~/Desktop/SCR/SCR_77/kod1$ ./detached.out
Main: creating thread 0
Main: creating thread 1
Thread 0 starting...
Main: creating thread 2
Thread 1 starting...
Main: creating thread 3
Main: program completed. Exiting.
Thread 3 starting...
Thread 2 starting...
Thread 3 done. Result = -3.153838e+06
Thread 0 done. Result = -3.153838e+06
Thread 1 done. Result = -3.153838e+06
Thread 2 done. Result = -3.153838e+06
```

Instrukcja join() sprawia, iż w miejscu jej wywołania program będzie czekał tak długo, aż dany wątek się nie zakończy. Domyślnie wątki pracują w tym trybie.

Jeżeli zostanie wywołana instrukcja detach() funkcja główna w której został stworzony wątek może zakończyć swoje działanie bez czekania, aż wątek się wykona. Daje nam to większą swobodę, ale o synchronizację musimy zadbać sami.

6. Operacje ze stosem

Przejrzyj, skompiluj i uruchom program bug2.c.

Co się dzieje? Dlaczego? Jak to naprawić?

Dla porównania uruchom program bug2fix.c.

Stos zostaje przepełniony poprzez alokacje zbyt dużej tablicy A. Rozwiązaniem może być alokacja za pomocą funkcji malloc() lub zmiana rozmiaru stosu wątku, jak zostało zaprezentowane w przykładnie bug2fix.c.

```
void *Hello(void *threadid)
{
   double* A;
   A=malloc(sizeof (double)*ARRAY_SIZE);
   int i;
   long tid;

   tid = (long)threadid;
   sleep(3);
   for (i=0; i<ARRAY_SIZE; i++)
   {
      A[i] = i * 1.0;
   }
   printf("%ld: Hello World!   %f\n", tid, A[ARRAY_SIZE-1]);
   free(A);
   pthread_exit(NULL);
}
```

```
pthread_attr_init(&attr);
stacksize = ARRAY_SIZE*sizeof(double) + MEGEXTRA;
pthread_attr_setstacksize (&attr, stacksize);
pthread_attr_getstacksize (&attr, &stacksize);
```