16.10.2023	Łukasz Oprych Gr.5 ITE	Temat: POSIX

Cel:

Zapoznanie się z operacjami na wątkach przy pomocy biblioteki pthreads.

Przebieg ćwiczenia:

Uzupełnienie skryptu zgodnie z poleceniem prowadzącego:

```
printf("watek glowny: tworzenie watku potomnego nr 1\n");
pthread_create(&tid, NULL, zadanie_watku, NULL);
sleep(2); // czas na uruchomienie watku
printf("\twatek glowny: wyslanie sygnalu zabicia watku\n");
pthread_cancel(tid);

pthread_join(tid, &wynik);
if (wynik == PTHREAD_CANCELED)
        printf("\twatek glowny: watek potomny zostal zabity\n");
else
printf("\twatek glowny: watek potomny NIE zostal zabity - blad\n");
```

Za pomocą funkcji pthread_create utworzono nowy wątek, przekazując NULL jako argument attr, co oznacza, że przyjęto "domyślne wartości" dla tego nowego wątku. ID tego nowego wątku w systemie zostało zapisane do zmiennej tid. Następnie, przy użyciu funkcji pthread_join, czekało się aktywnie na zakończenie działania tego wątku. Wynik zwracany przez ten wątek został zapisany w zmiennej wynik.

```
zmienna_wspolna = 0;

printf("watek glowny: tworzenie watku potomnego nr 2\n");
pthread_create(&tid, NULL, zadanie_watku, NULL);
sleep(2); // czas na uruchomienie watku
printf("\twatek glowny: odlaczenie watku potomnego\n");
pthread_detach(tid);

printf("\twatek glowny: wyslanie sygnalu zabicia watku odlaczonego\n");
pthread_cancel(tid);

printf("\twatek glowny: czy watek potomny zostal zabity \n");
printf("\twatek glowny: sprawdzanie wartosci zmiennej wspolnej\n");
for(i=0;i<10;i++){
        sleep(1);
if(zmienna_wspolna!=0) break;
}

if (zmienna_wspolna==0)
        printf("\twatek glowny: odlaczony watek potomny PRAWDOPODOBNIE zostal zabity\n");
else
printf("\twatek glowny: odlaczony watek potomny PRAWDOPODOBNIE NIE zostal zabity\n");
```

Utworzono nowy wątek i odłączono go od wątku głównego za pomocą pthread_detach. Następnie, w pętli, co każdą iterację, wątek główny był zawieszany na 1 sekundę, co ostatecznie sumowało się do łącznego oczekiwania przez 10 sekund na zakończenie pracy tego oddzielonego wątku. Zatrzymywano ten proces, gdy "zmienna_wspolna" przyjmowała wartość jeden. To oznaczało, że wątek nie odebrał sygnału zatrzymania i zakończył swoje działanie bez wymuszenia. Po zakończeniu pętli dodawano informację, która na podstawie wartości "zmienna_wspolna" określała, czy wątek został

przerwany, czy nie. Wartość równa 0 wskazywała na przerwanie wątku, podczas gdy wartość różna od zera sugerowała, że wątek nie otrzymał sygnału zatrzymania.

```
pthread_attr_init(&attr);
pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_DETACHED);

printf("watek glowny: tworzenie odlaczonego watku potomnego nr 3\n");
pthread_create(&tid, &attr, zadanie_watku, NULL);
pthread_attr_destroy(&attr);

printf("\twatek glowny: koniec pracy, watek odlaczony pracuje dalej\n");
pthread_exit(NULL);
```

Następnie zaczęto od inicjalizacji domyślnych ustawień atrybutów wątku w zmiennej "attr" i ustawiono atrybut, który odpowiada za stworzenie wątku jako "odłączony" podczas jego tworzenia. Następnie przekazano zmienną z tymi atrybutami jako drugi argument do funkcji pthread_create, co skutkowało stworzeniem wątku, który był już odłączony od wątku głównego od samego początku. Aby utrzymać spójność, zniszczono zmienną, która przechowywała te atrybuty, gdy już nie były potrzebne.

Następnie uruchomiono program:

```
loprych@loprych-VirtualBox:~/Desktop/PR_lab/lab_3/zad_1$ ./pthreads_detach_kill
watek glowny: tworzenie watku potomnego nr 1
        watek potomny: uniemozliwione zabicie
        watek glowny: wyslanie sygnalu zabicia watku
        watek potomny: umozliwienie zabicia
        watek glowny: watek potomny zostal zabity
watek glowny: tworzenie watku potomnego nr 2
        watek potomny: uniemozliwione zabicie
        watek glowny: odlaczenie watku potomnego
        watek glowny: wyslanie sygnalu zabicia watku odlaczonego
        watek glowny: czy watek potomny zostal zabity
        watek glowny: sprawdzanie wartosci zmiennej wspolnej
        watek potomny: umozliwienie zabicia
        watek glowny: odlaczony watek potomny PRAWDOPODOBNIE zostal zabity
watek glowny: tworzenie odlaczonego watku potomnego nr 3
        watek glowny: koniec pracy, watek odlaczony pracuje dalej
        watek potomny: uniemozliwione zabicie
        watek potomny: umozliwienie zabicia
        watek potomny: zmiana wartosci zmiennej wspolnej
```

Następnie utworzono katalog zad_2 wraz z nowym plikiem źródłowym.

```c
#define LICZBA_W_MAX 4

void* zadanie_watku(void* arg_wsk) {
    int moj_arg = *((int*)arg_wsk);
    pthread_t x = pthread_self();
    printf("ID systemowy: %d i identyfikator = %ld\n", moj_arg, x);
    return NULL;
}

int main() {
    pthread_t threads[LICZBA_W_MAX];
    int indeksy[LICZBA_W_MAX];
    int p = LICZBA_W_MAX;

    printf("Dzialam\n");

    for (int i = 0; i < p; i++) {
        indeksy[i] = i;
    }

    for (int i = 0; i < p; i++) {
        pthread_create(&threads[i], NULL, zadanie_watku, (void*)&indeksy[i]);
    }
    for (int i = 0; i < p; i++) {
        pthread_join(threads[i], NULL);
    }

    pthread_exit(NULL);
    printf("Dzialam\n");
}
```

W funkcji „zadanie_watku” uzyskiwany jest identyfikator danego wątku, przy użyciu pthread_self i przechowywany w zmiennej x. Na koniec funkcja wypisuje przechowywany numer wątku oraz indetyfikator systemowy wątku.

Liczba wątków została określona za pomocą makra preprocesora, co umożliwia łatwą modyfikację liczby elementów w tablicy „indeksy”. Przypisywaliśmy kolejne wartości do poszczególnych komórek w tej tablicy, zaczynając od 0 i kontynuując aż do wartości o jeden mniejszej niż LICZBA_W_MAX. Dopiero po wykonaniu tego procesu tworzono wątki, a następnie oczekiwano, aż wszystkie zakończą swoje działanie. Na koniec używając pthread_exit kończymy działanie wątku głównego.

Wynik programu:

```
loprych@loprych-VirtualBox:~/Desktop/PR_lab/lab_3/zad_2$ ./program
Dzialam
ID systemowy: 0 i identyfikator = 140593328027200
ID systemowy: 2 i identyfikator = 140593311241792
ID systemowy: 1 i identyfikator = 140593319634496
ID systemowy: 3 i identyfikator = 140593302849088
loprych@loprych-VirtualBox:~/Desktop/PR_lab/lab_3/zad_2$
```

Wnioski:

Wątki, mogą pracować w trybie detached, czyli odłączonym oraz joinable – standardowym. Wątek standardowy, wykonuje swoje polecenia i zakańcza po tym swoje działanie, można też to wymusić za pomocą funkcji pthread_cancel. W celu zabezpieczenia wątku standardowego przed zakończeniem jego działaniem możemy zastosować funkcję pthread_cancel_disable, co blokuje wykonywanie funkcji pthread_cancel. Tworzenie wątków odłączonych jest praktyczne w sytuacjach, gdy nie mamy zamiaru oczekiwać na ich zakończenie za pomocą funkcji pthread_join. Wątek odłączony samodzielnie zwalnia zasoby po swoim zakończeniu, podczas gdy wątek nieodłączony wymaga, aby inny wątek "dołączył" do niego, aby te zasoby zostały zwolnione.