Przemysław Szymoniak

Przetwarzanie współbieżne. Programowanie równoległe i rozproszone.

Sprawozdanie z laboratorium 3.

**Pthread** – jest to biblioteka wykorzystywana głównie w systemach z rodziny unix/linux jest implementacja wykorzystującą standard POSIX, dostarcza potrzebne mechanizmy do tworzenia, usuwania i zarządzania wątkami.

Celem laboratorium było nabycie praktycznej umiejętności manipulowania wątkami, przetestowanie mechanizmu przesyłania argumentów do wątków oraz poznanie funkcjonowania obiektów określających atrybuty wątku.

W ramach zajęć zrealizowałem następujące kroki:

* Pobranie pliku „**pthreads\_detach\_kill.c**” wraz z uzupełnieniem kodu programu zgodnie z opisem zawartym w skrypcie:

|  |  |
| --- | --- |
| **int main()**  **{**  **pthread\_t tid;**  **pthread\_attr\_t attr;**  **void \*wynik;**  **int i;**  **printf("glowny: tworzenie watku potomnego nr 1\n");**  **pthread\_create(&tid, NULL, zadanie\_watku, NULL);**  **sleep(2); // czas na uruchomienie watku**  **printf("\t glowny: wyslanie sygnalu zabicia watku\n");**  **pthread\_cancel(tid);**  **pthread\_join(tid, &wynik);**  **if (wynik == PTHREAD\_CANCELED)**  **printf("\t glowny: watek potomny został zabity\n");**  **else**  **printf("\t glowny: potomny NIE zostal zabity blad");**  **zmienna\_wspolna = 0;**  **printf(„glowny: tworzenie watku potomnego nr 2\n");**  **pthread\_create(&tid, NULL, zadanie\_watku, NULL);**  **sleep(2); // czas na uruchomienie watku** | **printf("\tglowny: odlaczenie watku potomnego\n");**    **pthread\_detach(tid);**  **printf("\tglowny: wyslanie sygnalu zabicia watku odlaczonego\n");**  **pthread\_cancel(tid);**  **printf("\tglowny: czy watek potomny zostal zabity \n");**  **printf("\tglowny: sprawdzanie war. 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");**    **printf("glowny: tworzenie odlaczonego watku potomnego nr 3\n");**  **pthread\_create(&tid, &attr, zadanie\_watku, NULL);**    **printf("\tglowny: koniec pracy, watek odlaczony pracuje dalej\n");**  **pthread\_exit(NULL);**  **}** |

* Kompilacja programu poleceniem **gcc –pthread –c** **pthreads\_detach\_kill.c**
* Utworzenie nowego programu w którym do procedury wątku przesyłany jest identyfikator wątku wypisując go na ekranie razem z identyfikatorem systemowym, kod:

**void \* zadanie\_watku (void \* arg\_wsk){  
printf("identyfikator: %d \n wskaznik: %d \n\n",pthread\_self(), \*((int \*)arg\_wsk));**

**pthread\_setcancelstate(PTHREAD\_CANCEL\_ENABLE, NULL);  
pthread\_testcancel();  
sleep(2);  
return(NULL);  
}**

|  |  |
| --- | --- |
| **main(){**  **pthread\_t tid[10];**  **void \*wynik;**  **int i=0;**  **int id[10];**  **for(i=0;i<10;i++){**  **id[i]=i;**  **pthread\_create(&tid[i], NULL, zadanie\_watku, &id[i]);**  **}** | **for(i=0;i<10;i++){**  **pthread\_cancel(tid[i]);**  **pthread\_join(tid[i], &wynik);**  **if (wynik == PTHREAD\_CANCELED)**  **printf("watek potomny zostal zabity\n");**  **else**  **printf("watek potomny NIE zostal zabity blad\n");**  **}**  **pthread\_exit(NULL);**  **}** |

* Uruchomienie, testowanie kodu. Stworzenie nowego kody programu w którym jako argument jest przesyłana struktura zawierająca 3 pola, modyfikująca wartości lokalne w procedurze wątku:

|  |  |
| --- | --- |
| **struct watek{**  **int ID;**  **int dzien;**  **int miesiac;**  **};**  **void \* zadanie\_watku (void \* arg\_wsk)**  **{**  **struct watek bufor = \*((struct watek\*)arg\_wsk);**  **int id =bufor.ID;**  **int dzien = bufor.dzien;**  **int miesiac = bufor.miesiac;**  **id = rand();**  **dzien=rand();**  **miesiac=rand();**    **printf("structura: identyfikator: %d \n dzien: %d \n miesiac: %d\n\n", bufor.ID, bufor.dzien, bufor.miesiac);**  **printf("lokal: identyfikator: %d \n dzien: %d \n miesiac: %d\n\n",id, dzien, miesiac);**  **pthread\_setcancelstate(PTHREAD\_CANCEL\_ENABLE, NULL);**  **pthread\_testcancel();**  **sleep(2);**  **return(NULL);**  **}** | **main(){**  **srand(time(0));**  **pthread\_t tid[10];**  **void \*wynik;**  **int i=0;**  **struct watek id;**  **for(i=0;i<2;i++){**  **id.ID = 10;**  **id.dzien = 50;**  **id.miesiac = 70;**  **pthread\_create(&tid[i], NULL, zadanie\_watku, &id);**  **}**  **printf("MAIN: identyfikator: %d \n dzien: %d \n miesiac: %d\n\n", id.ID, id.dzien, id.miesiac);**  **for(i=0;i<2;i++){**  **pthread\_join(tid[i], &wynik);**  **if (wynik == PTHREAD\_CANCELED)**  **printf("\watek potomny zostal zabity\n");**  **else**  **printf("\watek potomny NIE zostal zabity blad\n");**  **}**  **printf("MAIN: identyfikator: %d \n dzien: %d \n miesiac: %d\n\n", id.ID, id.dzien, id.miesiac);**  **pthread\_exit(NULL);**  **}** |

Wnioski:

* Wykorzystywanie możliwości tworzenia procesów oraz wątków pozwala znacząco zwiększyć wydajność aplikacji dzięki podzieleniu pewnych zadań na parę mniejszych wykonywanych równolegle
* Identyfikatory wątków wyświetlane były w różnej kolejności, co sugeruje że wątki nie są wykonywane chronologicznie
* Aby uzyskać rosnącą listę numeracji identyfikatorów wyświetlanych przez wątki należało opóźnić wykonywanie poszczególnych wątków przez użycie funkcji **„sleep()”**
* Przesyłanie argumentów jako referencja do zmiennej lub struktury wymaga wykonania rzutowania z adresu na typ wysyłanych danych pozwalając jednocześnie zmniejszyć czas oraz obszar pamięci zajmowanej  
  w stosunku do przesyłania argumentów przez wartość
* Program poprawnie wypisywał zmienne lokalne modyfikowane a po zakończeniu pętli zmienna globalna zostawała nie zmieniona
* Przed wypisaniem zmiennej globalnej należy użyć funkcji „**pthread\_join()”**