|  |  |  |
| --- | --- | --- |
| |  | | --- | | **INSTYTUT TELEINFORMATYKI I AUTOMATYKI**  **Wydział Cybernetyki WAT** | | |
| |  | | --- | | Przedmiot: SYSTEMY OPERACYJNE  SPRAWOZDANIE Z ĆWICZENIA LABORATORYJNEGO NR 1 | | |
| **Temat**: Synchronizacja procesów i wątków II | |
| **Wykonał:**  xxxxx | **Data wykonania ćwiczenia:**  xxxxx  **Prowadzący ćwiczenie:**  xxxxx |

1. **Treść zadania laboratoryjnego**

Obraz zawierający tekst, osoba

Opis wygenerowany automatycznie

Do synchronizacji wątków wykorzystać mechanizm wzajemnego wykluczania przy pomocy implementacji mutexów

1. **Opis rozwiązania, komentarze, wnioski**

Program:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/msg.h>

#include <sys/sem.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <time.h>

#include <signal.h>

#include <pthread.h>

#define steps 14

pthread\_mutex\_t mt1, mt2, mtp;

pthread\_t t1, t2, tp;

int i=1;

char c[1];

void \*Threadp(void\* v){ //Producent thread

    while(i < steps){

        pthread\_mutex\_lock(&mtp); //Lock p if available

        c[0] = 'A' + (random() % 26);

        printf("Thread p - step: %d\t Drawn character %c\n", i, c[0]);

        i++;

        pthread\_mutex\_unlock(&mt2); //Unlocking 2

    }

    //printf("End of thread p\n");

    pthread\_exit(0);

}

void \*Thread1(void\* v){ //Consument 1 thread

    while(i < steps){

        pthread\_mutex\_lock(&mt1); //Lock 1 if available

        printf("Thread 1 - step: %d\t Imported character %c\n", i, c[0]);

        i++;

        pthread\_mutex\_unlock(&mtp); //Unlocking p

    }

    //printf("End of thread 1\n");

    pthread\_exit(0);

}

void \*Thread2(void\* v){ //Consument 2 thread

    while(i < steps){

        pthread\_mutex\_lock(&mt2); //Lock 2 if available

        printf("Thread 2 - step: %d\t Imported character %c\n", i, c[0]);

        i++;

        pthread\_mutex\_unlock(&mt1); //Unlocking 1

    }

    //printf("End of thread 2\n");

    pthread\_exit(0);

}

int main(){

    srand(time(NULL));

    printf("Inicjalizing threads\n");

    pthread\_mutex\_init(&mtp, NULL);

    pthread\_mutex\_init(&mt1, NULL);

    pthread\_mutex\_init(&mt2, NULL);

    printf("Locking mutex 1 and 2\n");

    pthread\_mutex\_lock(&mt1);

    pthread\_mutex\_lock(&mt2);

    printf("Calling threads\n");

    pthread\_create(&tp, NULL, Threadp, NULL);

    pthread\_create(&t1, NULL, Thread1, NULL);

    pthread\_create(&t2, NULL, Thread2, NULL);

    printf("Waiting for threads to finish\n");

    pthread\_join(tp, NULL);

    pthread\_join(t1, NULL);

    pthread\_join(t2, NULL);

    printf("Deleting threads from memory\n");

    pthread\_mutex\_destroy(&mtp);

    pthread\_mutex\_destroy(&mt1);

    pthread\_mutex\_destroy(&mt2);

    printf("End\n");

    return 0;

}

Kolejność działań:

1. Działanie wątku p (producenta)
   1. Producent losuje liczbę
   2. Zwalnia blokadę wątku 2
2. Działanie wątku 2
   1. Pobiera i wyświetla liczbę
   2. Zwalnia blokadę wątku 1
3. Działanie wątku 1
   1. Pobiera i wyświetla liczbę
   2. Zwalnia blokadę wątku p
4. Powrót do punktu 1 (pętla)

Chcąc wyznaczyć wątek rozpoczynający działanie zapętlonego działania należy zablokować wszystkie pozostałe wątki.

Program czeka do momentu zakończenia działań wszystkich wątków, a następnie czyści pamięć ze stworzonych mutexów.

1. **Wyniki uruchomienia programu**

Obraz zawierający tekst, tabliczka

Opis wygenerowany automatycznie