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

1. **Treść zadania laboratoryjnego**

Obraz zawierający tekst, osoba

Opis wygenerowany automatycznie

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

Program:

Program Producenta („p.c”):

#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 <string.h>

struct sembuf bufor;

void setSem(int semid, int semnum, int opt) {

  bufor.sem\_num = semnum;

  bufor.sem\_op = opt;

  bufor.sem\_flg = 0;

  if (semop(semid, &bufor, 1) == -1) {

    perror("Error with semaphore update\n");

    exit(1);

  }

};

void validateSem(int v) {

  if (v == -1) {

    perror("Error with semaphore value");

    exit(1);

  }

};

int main() {

  pid\_t k1, k2;

  if ((k1=fork()) == 0) { // K2

     execlp("./k2", "k2", (char \*)NULL);

  } else if ((k2 = fork()) == 0) { // K1

     execlp("./k1", "k1", (char \*)NULL);

  } else { // P

     int semid = semget(45287, 3, IPC\_CREAT|0600);

     if (semid == -1) {

       perror("Error with creating semaphore table\n");

       exit(1);

     }

     validateSem(semctl(semid, 0, SETVAL, (int)1));

     validateSem(semctl(semid, 1, SETVAL, (int)0));

     validateSem(semctl(semid, 2, SETVAL, (int)0));

     int shmid = shmget(45286, sizeof(char), IPC\_CREAT|0600);

     if (shmid == -1) {

       perror("Error with creating shared memory\n");

       exit(1);

     }

     char \*bufor = shmat(shmid, NULL, 0);

     if (bufor == NULL) {

       perror("Error with joining a shared memory segment\n");

       exit(1);

     };

     char c[2];

     int i;

     for (i=0; i<10; i++) {

       setSem(semid, 0, -1);

       c[0] = 'A' + (random() % 26);

       c[1] = '\0';

       strcpy(bufor, c);

       printf("P1 - save value: %s\n", bufor);

       if(i!=0) printf("\n");

       setSem(semid, 2, 1);

     }

     sleep(3);

     kill(k1, SIGINT);

     kill(k2, SIGINT);

     shmdt(bufor);

     shmctl(shmid, IPC\_RMID, NULL);

     semctl(semid, 1, IPC\_RMID);

  }

  return 0;

};

Program konsumenta 1 („k1.c”):

#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>

struct sembuf bufor;

void setSem(int semid, int semnum, int opt) {

  bufor.sem\_num = semnum;

  bufor.sem\_op = opt;

  bufor.sem\_flg = 0;

  if (semop(semid, &bufor, 1) == -1) {

    perror("Error with semaphore value\n");

    exit(1);

  }

};

int main() {

  printf("K1 start\n\n");

  int semid = semget(45287, 3, IPC\_CREAT|0600);

  if (semid == -1) {

    perror("Error with semaphore table connection\n");

    exit(1);

  }

  int shmid = shmget(45286, sizeof(char), IPC\_CREAT|0600);

  if (shmid == -1) {

    perror("Error with connecting shared memory\n");

    exit(1);

  }

  char \*bufor = shmat(shmid, NULL, 0);

  if (bufor == NULL) {

    perror("Error with joining a shared memory segment\n");

    exit(1);

  }

  while (1) {

    setSem(semid, 1, -1);

    sleep(1);

    printf("K1 - received value: %s\n", bufor);

    setSem(semid, 0, 1);

  }

  return 0;

};

Program konsumenta 2 („k2.c”):

#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>

struct sembuf bufor;

void setSem(int semid, int semnum, int opt) {

  bufor.sem\_num = semnum;

  bufor.sem\_op = opt;

  bufor.sem\_flg = 0;

  if (semop(semid, &bufor, 1) == -1) {

    perror("Error with semaphore value\n");

    exit(1);

  }

};

int main() {

  printf("K2 start\n");

  int semid = semget(45287, 3, IPC\_CREAT|0600);

  if (semid == -1) {

    perror("Error with semaphore table connection\n");

    exit(1);

  }

  int shmid = shmget(45286, sizeof(char), IPC\_CREAT|0600);

  if (shmid == -1) {

    perror("Error with connecting shared memory\n");

    exit(1);

  }

  char \*bufor = shmat(shmid, NULL, 0);

  if (bufor == NULL) {

    perror("Error with joining a shared memory segment\n");

    exit(1);

  }

  while (1) {

    setSem(semid, 2, -1);

    sleep(1);

    printf("K2 - received value: %s\n", bufor);

    setSem(semid, 1, 1);

  }

  return 0;

};

Kolejność działań:

1. Wykonuje plik producenta:
   1. Producent generuje znak
   2. Producent powołuje pamięć współdzieloną
   3. Konsument 2 wykonuje się:
      1. Łączy się z pamięcią współdzieloną
      2. pobiera ten znak
   4. Konsument 1 wykonuje się
      1. Łączy się z pamięcią współdzieloną
      2. pobiera ten znak
2. **Wyniki uruchomienia programu**

Obraz zawierający tekst, tabliczka

Opis wygenerowany automatycznie