Politechnika Świętokrzyska w Kielcach
Wydział Elektrotechniki, Automatyki i Informatyki

Programowanie Współbieżne

laboratorium

Temat:

Pamięć dzielona
(laboratorium nr 7)

Autor: **Mateusz Snoch**

Grupa: **3ID11A**

Data odbycia się laboratorium : **07.12.2017**

Data sporządzenia sprawozdania: **08.12.2017**

Przydatne dane:

shmget – tworzenie segmentu pamięci wspólnej lub otwieranie jej
shmdt – odłączenie pamięci wspólnej
shmctl – usunięcie segmentu pamięci dzielonej (z argumentem cmd jako IPC_RMID)
Ustalanie adresu:
jeżeli shmaddr = 0 to system sam wybiera adres
jeżeli shmaddr != 0 to przekazywany adres zależy od tego czy ustalony jest znacznik SHM_RND
jeżeli nie jest ustawiony to segment pamięci wspólnej będzie podłączony od adresu określonego przez argument shmaddr
jeżeli jest ustawiony to zacznie się od adresu zaokrąglonego w dół o wartość stałej SHMLBA (Lower Boundary Address)

Zadanie

Napisać program serwera który:
tworzy segment pamięci dzielonej o wielkości pozwalającej na zapis zmiennej typu int
tworzy dwa semafory: 1 opuszczony, 2 podniesiony
w pętli wykonuje: opuszcza semafor 2, odczekuje losowy czas od 0 do 1 sekundy, wpisuje kolejną (poczynając od 1) liczbę do zmiennej dzielonej, podnosi semafor 1
program powinien usuwać pamięć dzieloną i semafory po naciśnięciu ctrl^c
należy zadbać by tylko jeden program serwera tworzył semafory i pamięć (tryb exclusive)
Napisać program klienta który:
podłącza się do segmentów pamięci i semaforów
w pętli (10 razy) podnosi semafor 2, opuszcza semafor 1, odczytuje wartość z pamięci dzielonej, wyświetla wartość, odczekuje losowy czas od 0 do 1 sekundy
sprawdzić działanie kilku klientów naraz
czy istnieje niebezpieczeństwo, że dwóch klientów dostanie tą samą liczbę?

Kod serwera :

```
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>
#include<stdlib.h>
#include<stdio.h>
#include<sys/sem.h>
#include<signal.h>
#include <unistd.h>

#define PERMS 0666

int *shmp;
int semid;
int shmid;
int dzialanie=1;

static struct sembuf op_lock[1] = {
        1, -1, 0
```

```
};

static struct sembuf op_unlock[1] = {
        0, 1, 0
};

void blokuj(int semid)
{
        if (semop(semid, &op_lock[0], 1)<0)
                perror("blad lokowania semafora");
}

void odblokuj(int semid)
{
        if (semop(semid, &op_unlock[0], 1) < 0)
                perror("blad odlokowania semafora");
}

void obsluga_zakonczenia_serwera(int nr_sig)
{
        printf("\nDzialanie serwera konczone!\n");
        dzialanie=0;

        if(shmdt(shmp)<0) perror("blad odlaczenia pamieci dzielonej");
        if(semctl(semid,0,IPC_RMID,0)<0) perror("blad usuwania zbioru
semaforow");
        if(shmctl(shmid,IPC_RMID,NULL)<0) perror("blad usuwania pamieci
dzielonej");
        exit(1);
}

int main(){

        signal(SIGINT,obsluga_zakonczenia_serwera);
        srand(getpid());
        shmid= shmget(ftok("serwer.c",2), sizeof(int),IPC_CREAT | IPC_EXCL
| PERMS);
        if (shmid==-1){
                perror("blad tworzenia pamieci dzielonej");
                exit(0);
        }
        shmp=shmat(shmid,0,0);
        if(shmp==NULL){
                perror("blad dolaczania pamieci dzielonej");
                exit(0);
        }
        semid = semget(ftok("serwer.c",2), 2, IPC_CREAT | PERMS);
        if  (semid < 0){
                perror("blad tworzenia zioru semaforow");
                exit(0);
        }
        int liczba=1;
        while(dzialanie==1){
                blokuj(semid);
                int time=rand()%1000001;
                printf("Uspione przez: %d\n",time);
                usleep(time);
                *shmp=liczba;
                liczba++;
                odblokuj(semid);
        }
```

```
        return 0;
}
```

Kod clienta :

```
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>
#include<stdlib.h>
#include<stdio.h>
#include<sys/sem.h>
#include<signal.h>
#include <unistd.h>

#define PERMS 0666

int *shmp;
int semid;
int shmid;

static struct sembuf op_lock[1] = {
        0, -1, 0
};

static struct sembuf op_unlock[1] = {
            1, 1, 0
};

void blokuj(int semid)
{
        if (semop(semid, &op_lock[0], 1)<0)
                perror("blad lokowania semafora");
}

void odblokuj(int semid)
{
        if (semop(semid, &op_unlock[0], 1) < 0)
                perror("blad odlokowania semafora");
}

int main(){
        int liczba;
        int i;

        srand(getpid());
        shmid= shmget(ftok("serwer.c",2), sizeof(int), PERMS);
        if (shmid==-1){
                perror("blad tworzenia pamieci dzielonej");
                exit(0);
        }
        int *shmp=shmat(shmid,0,0);
        if(shmp==NULL){
                perror("blad dolaczania pamieci dzielonej");
                exit(0);
        }
        semid = semget(ftok("serwer.c",2), 2, IPC_CREAT | PERMS);
        if  (semid < 0){
                perror("blad tworzenia zioru semaforow");
                exit(0);
```

```
        }

        for (i=0;i<20;i++){
                odblokuj(semid);
                blokuj(semid);
                liczba=*shmp;
                printf("Liczba: %d\n",liczba);
                int time=rand()%1000001;
                usleep(time);
        }
        shmdt(shmp);
        return 0;
}
```

Wnioski

Na siódmych zajęciach laboratoryjnych mogłem wykorzystać wiedzę o semaforach do wykonania instrukcji z pamięci dzielonej. Oba tematy są bardzo połączone ze sobą, bowiem ta lekcja bazowała na poprzedniej. Zajęcia nie przysporzyły mi problemów.