Zaawansowana komunikacja międzyprocesowa - semafory i pamięć wspólna		
Dominik Wróbel	16 V 2019	Czw. 17:00

Spis treści

1 Pamięć dzielona i semafory

Pytanie 1

Pobierz, rozpakuj, przeanalizuj, skompiluj i uruchom plik: shm.zip
Program symuluje dwa konta, które wykonują przelewy.
Co się dzieje z sumą obydwu kont? Dlaczego?

W programie suma kont rośnie w czasie. Dzieje się tak dlatego, że transakcje wykonywane na kontach nie są atomiczne. Implementacja programu nie daje gwarancji, że zmniejszanie i zwiększanie stanu konta w procesie nastąpi po sobie. Po operacji zmniejszenia stanu konta w procesie, jego modyfikacji może dokonać drugi z procesów, co spowoduje, że kolejna operacja (zwiększanie konta) będzie działała na zmodyfikowanym już stanie konta.

Pytanie 2

Pobierz, rozpakuj, przeanalizuj, skompiluj i uruchom plik: sem.zip
Co się zmieniło?

W tym programie operacja zmniejszania i zwiększania stanu konta jest atomiczna, po operacji zmniejszenia stanu konta zawsze zajdzie operacja zwiększenia stanu konta w tym samym procesie. Dzieje się tak dlatego, że program ten stosuje semafory do kontroli dostępu do pamięci współdzielonej, dzięki czemu dany proces jest w stanie wykonać całą transakcję bez zmiany pamięci współdzielonej przez inny proces. Dzięki temu w programie stan sumy kont jest utrzymywany na stałym poziomie.

2 Zadanie do samodzielnego wykonania

Do wykonania wybrano zadanie **problem orangutanów**.

Zadanie

Nad głębokim kanionem, gdzieś w Ameryce Południowej, rozpięta jest lina. Używają jej orangutany by przekroczyć kanion. Lina wytrzymuje ciężar pięciu małp a dwa orangutany nie mogą jednocześnie przechodzić po niej z przeciwnych stron kanionu. Po wejściu na linę nie można zawrócić z drogi. Każda małpa oczekująca na przejście musi kiedyś zostać obsłużona.

Listing 1: orangutan.c

```
1
2   #include "line.h"
3 #include <stdlib.h>
4
5 int main(int argc, char* argv[]){
6
7   int shmid;
8   int side = -1;
9   line* struktura;
10  srand(time(0));
11
12  if((shmid = shmget(4444, sizeof(line), IPC_CREAT | 0600)) == -1 ){
13    perror("Utworzenie segmentu pamieci wspoldzielonej");
14    exit(EXIT_FAILURE);
15  }
16
```

```
17    if((struktura = (line *)shmat(shmid,NULL,0)) == -1){
18      perror("Przylaczanie segmentu pamieci wspoldzielonej");
19      exit(EXIT_FAILURE);
20    }
21  printf("\n");
22
23    while(1){
24      sleep(rand()%3);
25
26      if(struktura->number_of_orangutans < 2 && ( struktura->direction == side ||
        struktura->direction == 0)){
27        struktura->number_of_orangutans++;
28        struktura->direction = side;
29        printf("\nOrangutan %s idzie na strone %d", argv[1], side*-1);
30        sleep(3);
31        struktura->number_of_orangutans--;
32        if (struktura->number_of_orangutans == 0)
33    struktura->direction = 0;
34        side *= -1;
35        printf("\nOrangutan %s jest na stronie %d", argv[1], side);
36      }
37      else if( struktura->number_of_orangutans == 2)
38        printf("\nOrangutan %s czeka bo nie ma miejsca na linie", argv[1]);
39      else
40        printf("\nOrangutan %s czeka bo inny orangutani idzie w jego strone", argv[1]);
41    }
42
43
44
45    return 0;
46  }
47
```

Listing 2: line.h

```
1
2   #include <stdio.h>
3  #include <sys/types.h>
4  #include <sys/ipc.h>
5  #include <sys/shm.h>
6  #include <stdio.h>
7  #include <stdlib.h>
8
9  #define MAX_ORANGUTANS 20
10
11 typedef struct{
12    int number_of_orangutans; /* 0-5 */
13    int left_fifo[20];          /* if conflict, first in fifo */
14    int right_fifo[20];
15    int fifo_priority;         /* if conflict -1-left first 1- right first */
16    int direction;             /* -1- from left to right 1-from right to left 0-free*/
17 } line;
18
```