Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное учреждение высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

Институт «Электронных и информационных систем»

Кафедра «Информационных систем и технологий»

**Семафоры в UNIX как средство синхронизации процессов**

Лабораторная работа №6 по учебной дисциплине «Операционные системы»

Отчёт

Принял преподаватель:

\_\_\_\_\_\_\_\_\_ Ананьев В. В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2021 г.

Выполнил студент группы 9091:

\_\_\_\_\_\_\_ Ковалев А.Д.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2021 г.

Великий Новгород

2021

**Цель лабораторной работы**

Цель работы: познакомиться с механизмом синхронизации процессов в UNIX.

**Исходный текст программы**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <sys/sem.h>

#include <sys/shm.h>

#include <string.h>

#include <time.h>

void sem(int semId, int n, int d)

{

struct sembuf op;

op.sem\_op = d;

op.sem\_flg = 0;

op.sem\_num = n;

semop(semId, &op, 1);

}

void unlockSem(int semId, char \*check, int n)

{

sem(semId, n, 1);

check[n] = 0;

}

int lockSem(int semId, char \*check, int n)

{

int isBusy = check[n];

sem(semId, n, -1);

check[n] = 1;

return isBusy;

}

void sort(int semId, int memId, char checkMemOffset, const size\_t n)

{

int \*nums = (int \*)shmat(memId, 0, 0);

char \*checks = shmat(memId, 0, 0) + checkMemOffset;

for (int i = 0; i < n; i++)

{

int minInd = i;

for (int j = i + 1; j < n; j++)

{

lockSem(semId, checks, i);

lockSem(semId, checks, j);

if (nums[j] < nums[minInd])

{

minInd = j;

}

unlockSem(semId, checks, i);

unlockSem(semId, checks, j);

}

if (i != minInd)

{

lockSem(semId, checks, i);

lockSem(semId, checks, minInd);

int t = nums[i];

nums[i] = nums[minInd];

nums[minInd] = t;

unlockSem(semId, checks, i);

unlockSem(semId, checks, minInd);

}

}

}

void RandomNumbers(int \*nums, int n, int min, int max)

{

srand((unsigned)(time(0)));

for (int i = 0; i < n; i++)

{

nums[i] = min + rand() % (max - min + 1);

}

}

void PrintNumbers(int \*nums, int n)

{

for (int i = 0; i < n; i++)

{

printf("%d ", nums[i]);

}

printf("\n");

}

int main(int argc, char \*argv[])

{

const int N = atoi(argv[1]);

const int min = atoi(argv[2]);

const int max = atoi(argv[3]);

int memId = shmget(IPC\_PRIVATE, sizeof(int) \* N + N, 0600 | IPC\_CREAT | IPC\_EXCL);

int checkMemOffset = sizeof(int) \* N;

int semId = semget(IPC\_PRIVATE, N, 0600 | IPC\_CREAT);

int \*numbers = (int \*)shmat(memId, 0, 0);

char \*checks = shmat(memId, 0, 0) + checkMemOffset;

RandomNumbers(numbers, N, min, max);

PrintNumbers(numbers, N);

for (int i = 0; i < N; i++)

{

unlockSem(semId, checks, i);

}

int childId = fork();

if (childId == 0)

{

sort(semId, memId, checkMemOffset, N);

}

else

{

int i = 0;

int status;

do

{

printf("%d: ", i);

for (int j = 0; j < N; j++)

{

if (lockSem(semId, checks, j))

{

printf("[%d] ", numbers[j]);

}

else

{

printf("%d ", numbers[j]);

}

fflush(stdout);

unlockSem(semId, checks, j);

}

printf("\r\n");

status = waitpid(childId, NULL, WNOHANG);

i++;

} while (!status);

printf("Sort finished \r\n");

PrintNumbers(numbers, N);

shmctl(memId, 0, IPC\_RMID);

semctl(semId, 0, IPC\_RMID);

}

}

**Результат выполнения программы**

>> make

gcc sema.c -o sema

./sema 10 1 100

88 65 46 78 7 84 63 29 48 2

0: 88 65 46 78 7 84 63 29 48 2

1: 88 65 46 78 7 84 63 29 48 2

2: 88 65 46 78 7 84 63 29 48 2

3: 88 65 46 78 7 84 63 29 48 2

4: 88 65 46 78 7 84 63 29 48 2

5: 88 65 46 78 7 84 63 29 48 2

6: 88 65 46 78 7 84 63 29 48 2

7: 2 65 46 78 7 84 63 29 48 88

8: 2 7 46 78 65 84 63 29 48 88

9: 2 7 29 78 65 84 63 46 48 88

10: 2 7 29 46 65 84 63 78 48 88

11: 2 7 29 46 48 84 63 78 65 88

12: 2 7 29 46 48 63 84 78 65 88

13: 2 7 29 46 48 63 65 78 84 88

Sort finished

2 7 29 46 48 63 65 78 84 88

**Вывод:**

В процессе выполнения лабораторной работы я на практике познакомился с работой семафоров в UNIX.