Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования "Новгородский государственный университет имени Ярослава Мудрого"

Кафедра «Информационных технологий и систем»

Дисциплина «Операционные системы»

Отчет по лабораторной работе

«Семафоры в UNIX как средство синхронизации процессов»

Выполнил студент группы 9091

\_\_\_\_\_\_\_\_\_\_\_\_\_/Прокофьев Максим Юрьевич/

Подпись ФИО

Принял преподаватель

\_\_\_\_\_\_\_\_\_\_\_\_/Ананьев Владислав Валерьевич/

Подпись ФИО

Великий Новгород

2021

**Цель лабораторной работы**

Цель работы: познакомиться с механизмом синхронизации процессов в UNIX.

**Исходный текст программы**

|  |
| --- |
| Lab06.c |
| #include <stdio.h>  #include <sys/shm.h>  #include <stdlib.h>  #include <unistd.h>  #include <sys/types.h>  #include <sys/stat.h>  #include <sys/wait.h>  #include <fcntl.h>  #include <sys/sem.h>  #include <string.h>  #include <time.h>  void initSemValue(int semId, int n)  {  struct sembuf op;  op.sem\_op = 1;  op.sem\_flg = 0;  op.sem\_num = n;  semop(semId, &op, 1);  }  void tryToGetSemOrWait(int semId, int n)  {  struct sembuf op;  op.sem\_op = -1;  op.sem\_flg = 0;  op.sem\_num = n;  semop(semId, &op, 1);  }  void releaseSem(int semId, int n)  {  initSemValue(semId, 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, 4 \* N, 0600 | IPC\_CREAT | IPC\_EXCL);  if (memId <= 0)  {  printf("error with shmget()\n");  return -1;  }  int\* memNumbers = (int \*)shmat(memId, 0, 0);  if (NULL == memNumbers)  {  printf("error with shmat()\n");  return -2;  }  printf("Initial data: \n");  srand((unsigned)(time(0)));  for (int i = 0; i < N; i++)  {  memNumbers[i] = min + rand() % (max - min + 1);  printf("%d ", memNumbers[i]);  }  printf("\n");  const size\_t semCount = N;  int semId = semget(IPC\_PRIVATE, semCount, 0600 | IPC\_CREAT);  if (semId < 0)  {  perror("error with semget()");  return -1;  }  else  {  printf("semaphore set id = %i\n", semId);  }  pid\_t childId = fork();  if (childId < 0)  {  perror("error with fork()\n");  }  else if (childId > 0)  {  for (int i = 0; i < N; ++i)  {  printf("Parent: Freeing semaphore %i\n", i);  releaseSem(semId, i);  }  int status = 0;  int m = 1;  while (waitpid(-1, &status, 0 | WNOHANG) == 0)  {  printf("Step %i: ", m);  for (int i = 0; i < N; i++)  {  tryToGetSemOrWait(semId, i);  printf("%d ", memNumbers[i]);  releaseSem(semId, i);  }  printf("\n");  m++;  }  printf("\nSort completed\n");  for (int i = 0; i < N; i++)  {  tryToGetSemOrWait(semId, i);  printf("%d\n", memNumbers[i]);  releaseSem(semId, i);  }  shmctl(memId, IPC\_RMID, NULL);  semctl(semId, IPC\_RMID, 0);  }  else if (childId == 0)  {  int key = 0;  int first = 0;  for (int second = 1; second < N; second++)  {  tryToGetSemOrWait(semId, second);  key = memNumbers[second];  releaseSem(semId, second);  first = second - 1;  while (first>=0 && memNumbers[first]>key)  {  tryToGetSemOrWait(semId, first);  tryToGetSemOrWait(semId, first+1);  memNumbers[first + 1] = memNumbers[first];  releaseSem(semId, first);  releaseSem(semId, first+1);  first = first - 1;  tryToGetSemOrWait(semId, first + 1);  memNumbers[first + 1] = key;  releaseSem(semId, first + 1);  }  }  exit(0);  }  } |

**Результат выполнения программы**

|  |
| --- |
| Вывод в терминале |
| >> make  gcc lab06.c -o lab06  ./program 5 1 30  Initial data 26 20 27 15 17  semaphore set id = 10  Parent: Freeing semaphore  Parent: Freeing semaphore 1  Parent: Freeing semaphore 2  Parent: Freeing semaphore  Parent: Freeing semaphore 4  Step 1: 26 20 27 15 17  Step 2: 26 20 27 15 17  Step 3: 26 20 27 15 17  Step 4: 26 20 27 15 17  Step 5: 26 20 27 15 17  Step 6: 26 20 27 15 17  Step 7: 26 20 27 15 17  Step 8: 26 20 27 15 17  Step 9: 20 15 26 27 17  Step 10: 20 20 26 27 17  Step 11: 15 17 20 26 27  Step 12: 15 17 20 26 27  Step 13: 15 17 20 26 27  Step 14: 15 17 20 26 27  Step 15: 15 17 20 26 27  Step 16: 15 17 20 26 27  Sort completed  15  17  20  26  27 |

**Вывод**

Вывод: выполняя лабораторную работу, я познакомился с механизмом синхронизации процессов в UNIX.