Лабораторная работа №9. Семафоры в UNIX

Цели работы

1. Получение навыков при работе с семафорами.

Используемое программное обеспечение

При выполнении лабораторной работы будет использовано следующие про-граммное обеспечение: gcc – компилятор C/C++, vi – текстовый редактор.

Порядок выполнения лабораторной работы

1. Войти в систему Linux, указав имя и пароль, предварительно получив их у администратора.
2. В каталоге вашей группы создать файл программу с расширением *.cpp по варианту.
3. Написать программу на тему «Работа с семафорами в Unix».
4. Собрать программу.
5. Исполнить программу

Пример работы с семафорами Unix

Создание семафора

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <iostream>

/* В качестве ключа для семафора используем любое достаточно длинное число.
Оно будет выступать в качестве внешнего идентификатора под которым семафор будет известен для
любой программы которая пожелает его использовать. */

#define KEY (1492)

using namespace std;
void main()
{
int id; /* Внутренний идентификатор семафора, под которым он известен внутри       программы */

/*
Следующая переменная это аргумент для функции semctl(). Semctl() делает различные вещи с семафо
ром в зависимости от переданных аргументов. Мы будем использовать ее что бы удостовериться что из
начально значение семафора 0.
*/

    union semun {
        int val;
        struct semid_ds *buf;
        ushort * array;
    } argument;

   argument.val = 0;

/* Создаем семафор с внешним ключом KEY, если он уже не существует. Даем доступ всем.
*/

   id = semget(KEY, 1, 0666 | IPC_CREAT);

   /* Проверяем результат */

   if(id < 0)
   {
    //неудача
      cerr<<"Unable to obtain semaphore.\n";
      exit(0);
   }
```

```
/* На самом деле, мы получили массив семафоров. Второй параметр semget() был числом элементов в
массиве. В нашем случае – 1.
    /*

/* Устанавливаем значение семафора номер 0 в массиве. Передаем значение 0 */

   if( semctl(id, 0, SETVAL, argument) < 0)
   {
    //неудача
      cerr<<"Cannot set semaphore value.\n";
   }
   else
   {
    //успех
      err<<"Semaphore "<<KEY<<" initialized.\n";
   }
}
```

Уменьшение значения семафора

Эта операция также известна как P-operation. Приведенный ниже фрагмент программы ждет пока значение семафора будет больше 0 и сразу же “захватывает” его, уменьшая на единицу.

```
int id; /* Внутренний идентификатор семафора, под которым он известен    внутри программы */
   struct sembuf operations[1]; /* “Массив” опреаций применяемых к семафору*/

   int retval; /* Результат функции semop() */

/* Получаем внутренний идентификатор семафора с внешем идентификатором KEY. */
   id = semget(KEY, 1, 0666);
   if(id < 0)
   /* Семафор не существует */
   {
      cerr<<"Cannot find semaphore.\n";
      exit(0);
   }

   /* Выполняем P-operation. */

   /* Инициализируем sembuf структуру. */
   /* Какой семафор в массиве семафоров: */
    operations[0].sem_num = 0;
    /* Какая операция? Вычитаем 1 из значения семафора : */
    operations[0].sem_op = -1;
    /* Устанавливаем флаг для ожидания: */
    operations[0].sem_flg = 0;

    /* Выполняем действие */
    retval = semop(id, operations, 1);

    if(retval == 0)
    {
       cout<<"Successful P-operation.\n";
    }
    else
    {
       cout<<"P-operation did not succeed.\n";
    }
```

Увеличение значения семафора
Также известна как V-oprtaion. Принципиально мало, чем отличается от P-opderatioin.

Задание на лабораторную работу

Необходимо написать две программы. Первая программа должна после запуска ждать пока “освободиться”

(станет не нулевым) семафор. После этого эта программа должна вывести свой ID и время, сколько она ждала и завершиться. Допускается запуск нескольких экземпляров программы одновременно.
Втора программа должна прибавлять число, полученное из командой строки к значению семафора, который ожидает первая программа тем самым, возобновляя выполнение заданного количества экземпляров первой программы.

Требования к отчёту по лабораторной работе

Отчёт должен содержать:

1. Титульный лист
2. Вариант задания
3. Блок-схему и листинг программы.
4. Вывод