Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Курсовой проект по курсу
«Операционные системы»

ДИНАМИЧЕСКИЕ БИБЛИОТЕКИ

Студент: Злобина Валерия Вадимовна
Группа: М8О–208Б–21
Вариант: 20
Преподаватель: Соколов Андрей Алексеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022.

Цель курсового проекта

1. Приобретение практических навыков в использовании знаний, полученных в течении курса
2. Проведение исследования в выбранной предметной области
3.

Задание

Необходимо спроектировать и реализовать программный прототип в соответствии с выбранным вариантом. Произвести анализ и сделать вывод на основании данных, полученных при работе программного прототипа.

Вариант

Необходимо написать 3-и программы. Далее будем обозначать эти программы A, B, C. Программа A принимает из стандартного потока ввода строки, а далее их отправляет программе C. Отправка строк должна производится построчно. Программа C печатает в стандартный вывод, полученную строку от программы A. После получения программа C отправляет программе A сообщение о том, что строка получена. До тех пор пока программа A не примет «сообщение о получение строки» от программы C, она не может отправлять следующую строку программе C. Программа B пишет в стандартный вывод количество отправленных символов программой A и количество принятых символов программой C. Данную информацию программа B получает от программ A и C соответственно. Способ организация межпроцессорного взаимодействия выбирает студент.

Общие сведения о программе

Программа состоит из четырёх файлов: program_A.cpp, program_B.cpp, program_C.cpp и main.cpp. Компиляция производится с помощью Makefile.

Общий метод и алгоритм решения:

В начале работы в main.cpp создаются два дочерних процесса для программ B и C, родительский процесс замещается программой A с помощью execl, сначала A с помощью getline считывает строку и передаёт в B количество считанных символов, а в C — количество считанных символов и саму строку посимвольно, затем B выводит количество введённых символов, полученное от A, а C выводит строку и передаёт B количество выведенных символов, после чего B выводит количество выведенных символов и цикл начинается заново. Межпроцессорное взаимодействие реализовано при помощи pipe и семафоров.

Исходный код:

main.cpp

```
#include <iostream>
#include <unistd.h>
```

```
#include <fcntl.h>
#include <semaphore.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <signal.h>
#include <stdarg.h>

int user_get(sem_t *semaphore)
{
   int s;
   sem_getvalue(semaphore, &s);
   return s;
}

void user_set(sem_t *semaphore, int n)
{
   while (user_get(semaphore) < n)
   {
      sem_post(semaphore);
   }
   while (user_get(semaphore) > n)
   {
      sem_wait(semaphore);
   }
}

int main()
{
   int fdAC[2];
   int fdAB[2];
   int fdBC[2];
   pipe(fdAC);
   pipe(fdAB);
   pipe(fdBC);
   sem_t* semA = sem_open("_semA", O_CREAT, 1);
   sem_t* semB = sem_open("_semB", O_CREAT, 0);
   sem_t* semC = sem_open("_semC", O_CREAT, 0);
   if ((semA == SEM_FAILED)||(semB == SEM_FAILED)||(semC == SEM_FAILED))
   {
      perror("sem_open");
      exit(1);
   }
   std::cout << "Введите строки:\n";
   pid_t C = fork();
   if (C == -1)
   {
      perror("fork");
      exit(2);
   }
   if (C == 0)
   {
      pid_t B = fork();
      if (B == -1)
      {
         perror("fork");
         exit(3);
      }
      if (B == 0)
      {
                              execl("B",   std::to_string(fdAB[0]).c_str(),   std::to_string(fdAB[1]).c_str(),
std::to_string(fdBC[0]).c_str(), std::to_string(fdBC[1]).c_str(), NULL);
      }
```

```
        else
        {
                                execl("C",   std::to_string(fdAC[0]).c_str(),   std::to_string(fdAC[1]).c_str(),
std::to_string(fdBC[0]).c_str(), std::to_string(fdBC[1]).c_str(), NULL);
        }
    }
    else
    {
                              execl("A",    std::to_string(fdAC[0]).c_str(),    std::to_string(fdAC[1]).c_str(),
std::to_string(fdAB[0]).c_str(), std::to_string(fdAB[1]).c_str(), NULL);
    }
    return 0;
}
```

program_A.cpp

```
#include <iostream>
#include <unistd.h>
#include <fcntl.h>
#include <semaphore.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <stdarg.h>
#include <signal.h>

int user_get(sem_t *semaphore)
{
    int s;
    sem_getvalue(semaphore, &s);
    return s;
}

void user_set(sem_t *semaphore, int n)
{
    while (user_get(semaphore) < n)
    {
        sem_post(semaphore);
    }
    while (user_get(semaphore) > n)
    {
        sem_wait(semaphore);
    }
}

int main(int args, char* argv[])
{
    int fdAC[2];
    fdAC[0] = atoi(argv[0]);
    fdAC[1] = atoi(argv[1]);
    int fdAB[2];
    fdAB[0] = atoi(argv[2]);
    fdAB[1] = atoi(argv[3]);
    sem_t* semA = sem_open("_semA", O_CREAT, 0777, 1);
    sem_t* semB = sem_open("_semB", O_CREAT, 0777, 0);
    sem_t* semC = sem_open("_semC", O_CREAT, 0777, 0);
    while(1)
    {
        std::string str;
        getline(std::cin, str);
        if (str == "END")
        {
            user_set(semA, 2);
```

```
            user_set(semB, 2);
            user_set(semC, 2);
            break;
        }
        int size = str.length();
        write(fdAC[1], &size, sizeof(int));
        write(fdAB[1], &size, sizeof(int));
        for (int i = 0; i < size; ++i)
        {
            write(fdAC[1], &str[i], sizeof(char));
        }
        user_set(semB, 1);
        user_set(semA, 0);
        while (user_get(semA) == 0)
        {
            continue;
        }
    }
    sem_close(semA);
    sem_destroy(semA);
    sem_close(semB);
    sem_destroy(semB);
    sem_close(semC);
    sem_destroy(semC);
    close(fdAC[0]);
    close(fdAC[1]);
    close(fdAB[0]);
    close(fdAB[1]);
    return 0;
}
```

program_B.cpp

```
#include <iostream>
#include <unistd.h>
#include <fcntl.h>
#include <semaphore.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <stdarg.h>
#include <signal.h>

int user_get(sem_t *semaphore)
{
    int s;
    sem_getvalue(semaphore, &s);
    return s;
}

void user_set(sem_t *semaphore, int n)
{
    while (user_get(semaphore) < n)
    {
        sem_post(semaphore);
    }
    while (user_get(semaphore) > n)
    {
        sem_wait(semaphore);
    }
}

int main(int args, char* argv[])
```

```
{
    int fdAB[2];
    fdAB[0] = atoi(argv[0]);
    fdAB[1] = atoi(argv[1]);
    int fdBC[2];
    fdBC[0] = atoi(argv[2]);
    fdBC[1] = atoi(argv[3]);
    sem_t* semA = sem_open("_semA", O_CREAT, 0777, 1);
    sem_t* semB = sem_open("_semB", O_CREAT, 0777, 0);
    sem_t* semC = sem_open("_semC", O_CREAT, 0777, 0);
    while (1)
    {
        while(user_get(semB) == 0)
        {
            continue;
        }
        if (user_get(semB) == 2)
        {
            break;
        }
        int size;
        read(fdAB[0], &size, sizeof(int));
        std::cout << "Отправлено символов: " << size << std::endl;
        user_set(semC, 1);
        user_set(semB, 0);
        while (user_get(semB) == 0)
        {
            continue;
        }
        if (user_get(semB) == 2)
        {
            break;
        }
        read(fdBC[0], &size, sizeof(int));
        std::cout << "Получено символов: " << size << std::endl;
        user_set(semA, 1);
        user_set(semB, 0);
        while(user_get(semB) == 0)
        {
            continue;
        }
        if (user_get(semB) == 2)
        {
            break;
        }
    }
    sem_close(semA);
    sem_close(semB);
    sem_close(semC);
    close(fdAB[0]);
    close(fdAB[1]);
    close(fdBC[0]);
    close(fdBC[1]);
    return 0;
}
```

program_C.cpp

```
#include <iostream>
#include <unistd.h>
#include <fcntl.h>
#include <semaphore.h>
```

```
#include <sys/types.h>
#include <sys/stat.h>
#include <stdarg.h>
#include <signal.h>

int user_get(sem_t *semaphore)
{
   int s;
   sem_getvalue(semaphore, &s);
   return s;
}

void user_set(sem_t *semaphore, int n)
{
   while (user_get(semaphore) < n)
   {
      sem_post(semaphore);
   }
   while (user_get(semaphore) > n)
   {
      sem_wait(semaphore);
   }
}

int main(int args, char* argv[])
{
   int fdAC[2];
   fdAC[0] = atoi(argv[0]);
   fdAC[1] = atoi(argv[1]);
   int fdBC[2];
   fdBC[0] = atoi(argv[2]);
   fdBC[1] = atoi(argv[3]);
   sem_t* semA = sem_open("_semA", O_CREAT, 0777, 1);
   sem_t* semB = sem_open("_semB", O_CREAT, 0777, 0);
   sem_t* semC = sem_open("_semC", O_CREAT, 0777, 0);
   while(1)
   {
      while(user_get(semC) == 0)
      {
         continue;
      }
      if (user_get(semC) == 2)
      {
         break;
      }
      int size;
      std::string str;
      read(fdAC[0], &size, sizeof(int));
      int t = 0;
      for (int i = 0; i < size; ++i)
      {
         char c;
         read(fdAC[0], &c, sizeof(char));
         str.push_back(c);
         t = i;
      }
      ++t;
      std::cout << str << std::endl;
      write(fdBC[1], &t, sizeof(int));
      user_set(semB, 1);
      user_set(semC, 0);
   }
```

```
    sem_close(semA);
    sem_close(semB);
    sem_close(semC);
    close(fdAC[0]);
    close(fdAC[1]);
    close(fdBC[0]);
    close(fdBC[1]);
    return 0;
}
```

Пример запуска программы:

```
valeria@valeria-Lenovo-ideapad-310-15IKB:~/OS/os_kp_1$ ./main
Введите строки:
12345
Отправлено символов: 5
12345
Получено символов: 5
strockadlaproverci123
Отправлено символов: 21
strockadlaproverci123
Получено символов: 21
d
Отправлено символов: 1
d
Получено символов: 1
```

Вывод:

При написании курсового проекта были использованы и закреплены навыки, полученные во время прохождения курса операционных систем.