|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **«Программное обеспечение ЭВМ и информационные технологии» (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 3 |

**Название:** Взаимодействие параллельных процессов

**Дисциплина:** Операционные системы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-52Б |  |  | В.А. Иванов |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Н.Ю. Рязанова |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2020

Задание 1

«Производство-потребление»

Листинг программы:

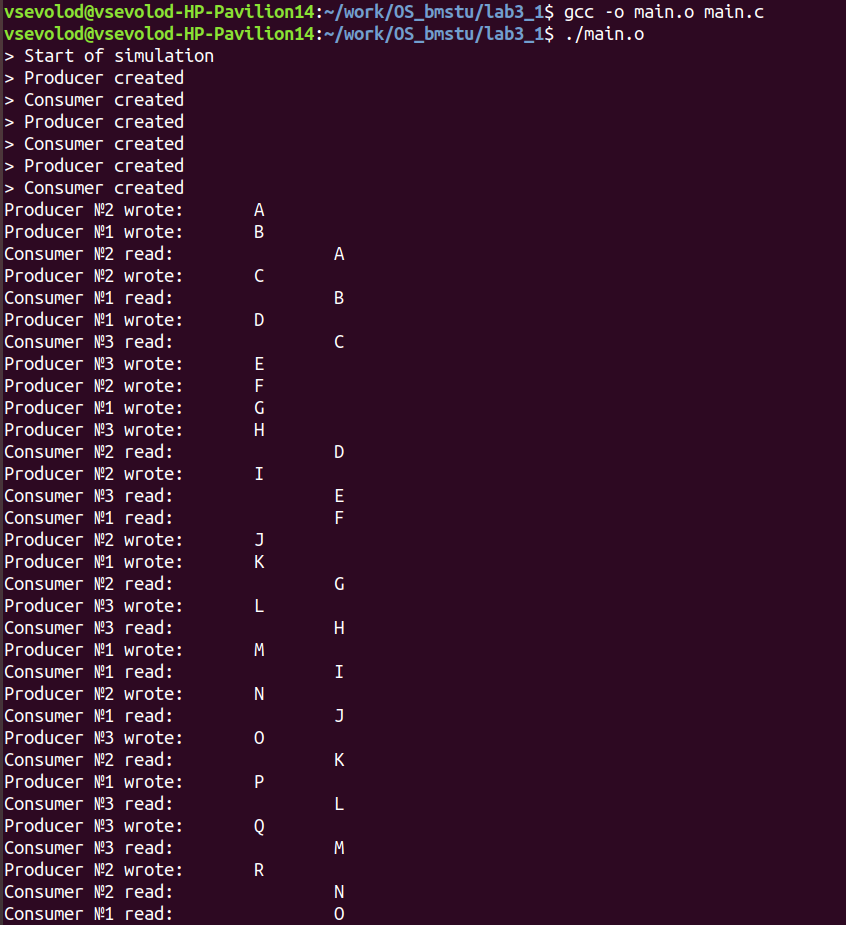
**main\_header.h**

|  |
| --- |
| #ifndef MAIN\_H  #define MAIN\_H  #include <sys/types.h>  #include <sys/ipc.h>  #include <sys/sem.h>  #include <sys/stat.h>  #include <sys/shm.h>  #include <unistd.h>  #include <sys/wait.h>  #include <stdio.h>  #include <stdlib.h>  #include <time.h>  #define FULL\_SEMN 0  #define EMPT\_SEMN 1  #define BIN\_SEMN 2  #define QUEUE\_SIZE 5  #define BUF\_SIZE 3 + QUEUE\_SIZE  #define PROC\_N 3  #endif // MAIN\_H |

**main.c**

|  |
| --- |
| #include "main\_header.h"  int rand\_dt()  {  return 1 + rand() % 3;  }  void prod\_func(size\_t b\_size, int isem\_descry, char\* buf, int my\_n)  {  srand(time(NULL) + (my\_n+1)\*100);  struct sembuf pre\_sem[2] = { {EMPT\_SEMN, -1, SEM\_UNDO},  {BIN\_SEMN, -1, SEM\_UNDO} };  struct sembuf post\_sem[2] = { {FULL\_SEMN, 1, SEM\_UNDO},  {BIN\_SEMN, 1, SEM\_UNDO} };  sleep(rand\_dt());  while (1)  {  if (semop(isem\_descry, pre\_sem, 2) == -1)  {  perror("semop");  exit(1);  }  buf[buf[2]] = buf[0];  printf("Producer №%d wrote:\t %c\n", my\_n+1, buf[0]);  if (++buf[2] >= b\_size)  buf[2] = 3;  if (++buf[0] > 'Z')  buf[0] = 'A';  if (semop(isem\_descry, post\_sem, 2) == -1)  {  perror("semop");  exit(1);  }  sleep(rand\_dt());  }    }  void cons\_func(size\_t b\_size, int isem\_descry, char\* buf, int my\_n)  {  srand(time(NULL) - (my\_n+1)\*100);  struct sembuf pre\_sem[2] = { {FULL\_SEMN, -1, SEM\_UNDO},  {BIN\_SEMN, -1, SEM\_UNDO} };  struct sembuf post\_sem[2] = { {EMPT\_SEMN, 1, SEM\_UNDO},  {BIN\_SEMN, 1, SEM\_UNDO} };  char cur\_letter;  sleep(rand\_dt());  while (1)  {  if (semop(isem\_descry, pre\_sem, 2) == -1)  {  perror("semop");  exit(1);  }  cur\_letter = buf[buf[1]];  printf("Consumer №%d read:\t\t %c\n", my\_n+1, cur\_letter);  if (++buf[1] >= b\_size)  buf[1] = 3;  if (semop(isem\_descry, post\_sem, 2) == -1)  {  perror("semop");  exit(1);  }  sleep(rand\_dt());  }    }  int main(void)  {  int perms = S\_IRWXU | S\_IRWXO | S\_IRWXG;  int isem\_descry = semget(IPC\_PRIVATE, 3, IPC\_CREAT | perms );  if (isem\_descry == -1)  {  perror("semget");  return 1;  }  int full\_ctl = semctl(isem\_descry, FULL\_SEMN, SETVAL, 0);  int empt\_ctl = semctl(isem\_descry, EMPT\_SEMN, SETVAL, QUEUE\_SIZE);  int bin\_ctl = semctl(isem\_descry, BIN\_SEMN, SETVAL, 1);  if (full\_ctl == -1 || empt\_ctl == -1 || bin\_ctl == -1)  {  perror("semctl");  return 1;  }  int mem\_id = shmget(IPC\_PRIVATE, (BUF\_SIZE)\*sizeof(char), IPC\_CREAT | perms);  if (mem\_id == -1)  {  perror("shmget");  return 1;  }  char\* addr = shmat(mem\_id, 0, 0);  if (addr == (char\*)(-1))  {  perror("shmat");  return 1;  }  addr[0] = 'A';  addr[1] = (char)3;  addr[2] = (char)3;  printf("> Start of simulation\n");  for (size\_t i=0; i<PROC\_N; i++)  {  pid\_t prod\_pid = fork();  switch (prod\_pid)  {  case -1:  perror("fork");  return 1;  case 0:  prod\_func(BUF\_SIZE, isem\_descry, addr, i);  return 0;  default:  printf("> Producer created\n");  break;  }  pid\_t cons\_pid = fork();  switch (cons\_pid)  {  case -1:  perror("fork");  return 1;  case 0:  cons\_func(BUF\_SIZE, isem\_descry, addr, i);  return 0;  default:  printf("> Consumer created\n");  break;  }  }  int status, pid;  for (size\_t i=0; i<PROC\_N\*2; i++)  {  pid = wait(&status);  if (pid == -1)  {  perror("wait");  return 1;  }  }  if (semctl(isem\_descry, 0, IPC\_RMID, 0) == -1)  {  perror("semctl");  return 1;  }  if (shmctl(mem\_id, IPC\_RMID, NULL) == -1)  {  perror("shmctl");  return 1;  }  if (shmdt(addr) == -1)  {  perror("shmdt");  return 1;  }  return 0;  } |

Пример работы:



Задание 2

«Читатели-писатели»

Листинг программы:

**main\_header.h**

|  |
| --- |
| #ifndef \_MAIN\_H  #define \_MAIN\_H  #include <sys/types.h>  #include <sys/ipc.h>  #include <sys/sem.h>  #include <sys/stat.h>  #include <sys/shm.h>  #include <unistd.h>  #include <sys/wait.h>  #include <stdio.h>  #include <stdlib.h>  #include <time.h>  #define ACT\_W\_SEMN 0  #define ACT\_R\_SEMN 1  #define WAI\_W\_SEMN 2  #define WAI\_R\_SEMN 3  #define WRITER\_N 2  #define READER\_N 5  #endif // \_MAIN\_H |

**main.c**

|  |
| --- |
| #include "main\_header.h"  int rand\_dt()  {  return 1 + rand() % 3;  }  void start\_read(int isem\_descry)  {  static struct sembuf wait\_sem[1] = {  {WAI\_R\_SEMN, 1, SEM\_UNDO} };  static struct sembuf act\_sem[4] = {  {ACT\_W\_SEMN, 0, SEM\_UNDO},  {WAI\_W\_SEMN, 0, SEM\_UNDO},  {ACT\_R\_SEMN, 1, SEM\_UNDO},  {WAI\_R\_SEMN,-1, SEM\_UNDO}};  if (semop(isem\_descry, wait\_sem, 1) == -1)  {  perror("semop");  exit(1);  }  if (semop(isem\_descry, act\_sem, 4) == -1)  {  perror("semop");  exit(1);  }  }  void stop\_read(int isem\_descry)  {  static struct sembuf act\_sem[1] = {  {ACT\_R\_SEMN,-1, SEM\_UNDO}};  if (semop(isem\_descry, act\_sem, 1) == -1)  {  perror("semop");  exit(1);  }  }  void start\_write(int isem\_descry)  {  static struct sembuf wait\_sem[1] = {  {WAI\_W\_SEMN, 1, SEM\_UNDO} };  static struct sembuf act\_sem[4] = {  {ACT\_W\_SEMN, 0, SEM\_UNDO},  {ACT\_R\_SEMN, 0, SEM\_UNDO},  {ACT\_W\_SEMN, 1, SEM\_UNDO},  {WAI\_W\_SEMN,-1, SEM\_UNDO}};  if (semop(isem\_descry, wait\_sem, 1) == -1)  {  perror("semop");  exit(1);  }  if (semop(isem\_descry, act\_sem, 4) == -1)  {  perror("semop");  exit(1);  }  }  void stop\_write(int isem\_descry)  {  static struct sembuf act\_sem[1] = {  {ACT\_W\_SEMN,-1, SEM\_UNDO}};  if (semop(isem\_descry, act\_sem, 1) == -1)  {  perror("semop");  exit(1);  }  }  void read\_func(int isem\_descry, char\* buf, int my\_n)  {  srand(time(NULL) + my\_n\*100);  while (1)  {  start\_read(isem\_descry);  printf("Reader №%d get:\t\t %c\n", my\_n, \*buf);  stop\_read(isem\_descry);  sleep(rand\_dt());  }  }  void write\_func(int isem\_descry, char\* buf, int my\_n)  {  srand(time(NULL) + my\_n\*10);  while (1)  {  start\_write(isem\_descry);  \*buf = (char)(rand() % ('z' - 'a') + 'a');  printf("Writer №%d send:\t %c\n", my\_n, \*buf);  stop\_write(isem\_descry);  sleep(rand\_dt());  }  }  int main(void)  {  int perms = S\_IRWXU | S\_IRWXO | S\_IRWXG;  int isem\_descry = semget(IPC\_PRIVATE, 4, IPC\_CREAT | perms );  if (isem\_descry == -1)  {  perror("semget");  return 1;  }  int ctl[4] = { 0, 0, 0, 0 };  ctl[0] = semctl(isem\_descry, ACT\_W\_SEMN, SETVAL, 0);  ctl[1] = semctl(isem\_descry, ACT\_R\_SEMN, SETVAL, 0);  ctl[2] = semctl(isem\_descry, WAI\_W\_SEMN, SETVAL, 0);  ctl[3] = semctl(isem\_descry, WAI\_R\_SEMN, SETVAL, 0);  for (size\_t i=0; i<4; i++)  if (ctl[i] == -1)  {  perror("semctl");  return 1;  }  int mem\_id = shmget(IPC\_PRIVATE, sizeof(char), IPC\_CREAT | perms);  if (mem\_id == -1)  {  perror("shmget");  return 1;  }  char\* addr = shmat(mem\_id, 0, 0);  if (addr == (char\*)(-1))  {  perror("shmat");  return 1;  }  \*addr = '!';  printf("> Start of simulation\n");  for (size\_t i=0; i<WRITER\_N; i++)  {  pid\_t prod\_pid = fork();  switch (prod\_pid)  {  case -1:  perror("fork");  return 1;  case 0:  write\_func(isem\_descry, addr, i);  return 0;  default:  printf("> Writer created\n");  break;  }  }  for (size\_t i=0; i<READER\_N; i++)  {  pid\_t prod\_pid = fork();  switch (prod\_pid)  {  case -1:  perror("fork");  return 1;  case 0:  read\_func(isem\_descry, addr, i);  return 0;  default:  printf("> Reader created\n");  break;  }  }  int status, pid;  for (size\_t i=0; i<WRITER\_N+READER\_N; i++)  {  pid = wait(&status);  if (pid == -1)  {  perror("wait");  return 1;  }  }  if (semctl(isem\_descry, 0, IPC\_RMID, 0) == -1)  {  perror("semctl");  return 1;  }  if (shmctl(mem\_id, IPC\_RMID, NULL) == -1)  {  perror("shmctl");  return 1;  }  if (shmdt(addr) == -1)  {  perror("shmdt");  return 1;  }  return 0;  } |

Пример работы:

