7 | 7 | 7

Контрольная работа №3.

Вариант 1. (ФИО: Хлопкин, 211гр.)

Задание 1. Написать две программы, общающиеся через IPC очередь

Программа - клиент запрашивает строки от пользователя (через stdin) и отсылает их через очередь программе-мастеру. Программа – мастер получает сообщения, генерируемые “клиентом” и выводит сообщения на stdout. Перед завершением своей работы “клиент” посылает сообщение-ограничитель (выбрать самостоятельно). После получения такого сообщения, мастер удаляет очередь.

Задание 2. Написать две программы, общающиеся через общую память.

Программа-“собиратель” должна запрашивать от пользователя количество собранных плодов со сказочных деревьев “Тыблоко”, “Ядро”, “КЛМН” и “Сэр” (четыре целочисленных числа). После чего программа-“собиратель” должна положить плоды (сохранить числа) на склад (в общей памяти), к которому будет иметь доступ “программа-волшебник”.

Программа-“волшебник” должна преобразовать волшебным способом в плоды некоторого дерева (на Ваш выбор) 60% плодов от остальных видов деревьев (прочитать сохранённые числа, провести вычисления и вывести результат на экран).

Требование: Синхронизацию “собирателя” и “волшебника” организовать через семафоры.

Задание 3. Написать программу.

У процесса-«отца» два «сына». Старший сын умеет выводить на экран предложение «Оценка за экзамен №K», где K – некоторое число. Младший сын умеет выводить сообщение «: N», где N – некоторое число. В совместной работе сыновья должны вывести в stdout:

Оценка за экзамен №1: 5
Оценка за экзамен №2: 4
Оценка за экзамен №3: 3
Оценка за экзамен №4: 2

- и только после этого умереть вместе с отцом.

Замечания: Механизм организации «смерти» произвольный. Синхронизацию работы «сыновей» реализовать с помощью каналов.

```
Задание 1.
struct TMessage //клиент
{ long type;
  char text[MSGMAX];
};
int main()
{
  key_t key;
  int msgid;
  struct TMessage message;
  char curr_str[MSGMAX];
  key = ftok("my_file", 45);
  msgid = msgget(key, 0666|IPC_CREAT);
  message.type = 1;
  do
  {
    i = -1;
    do
    {
      i++;
      curr_str[i] = fgetc(stdin);
    }
    while (curr_str[i] != '\n')
    curr_str[i] = '\0';
    strcpy(message.text, curr_str);
    msgsnd(msgid, &message, strlen(message.
                                   text)+1, 0);
  }
  while (strcmp(curr_str, "end") != 0)
  return 0;
}

// продолжение далее
```

+

```
// мастер
struct TMessage
{
    long type;
    char text[MSGMAX];
};
int main()
{
    key_t key;          struct TMessage message;
    int msgid;          char curr_str[MSGMAX];
    key = ftok("my_file", 45);
    msgid = msgget(key, 0666|IPC_CREAT);
    do
    {
        msgrcv(msgid, &message, MSGMAX, 0, 0);
        strcpy(curr_str, message.text);
        if(strcmp(curr_str, "end") != 0)
            printf("%s\n", curr_str);
    }
    while(strcmp(curr_str, "end") != 0)
    msgctl(msgid, IPC_RMID, NULL);
    return 0;
}
```

Задание 3;

```
int main()
{
    int fd1[2];     int status;
    int fd2[2];     int i;
    pipe(fd1);
    pipe(fd2);
    if(fork() == 0)  // 1 сын
    {
        close(fd1[0]); close(fd2[1]);
        for(i=0; i<4;)
        {
            printf("Оценка за экзамен №%d:", i+1);
            write(fd1[1], &i, sizeof(int));
            read(fd2[0], &i, sizeof(int));
        }
        close(fd1[1]); close(fd2[0]);
        exit(0);
    }
    else if(fork() == 0) // 2 сын
    {
        close(fd1[1]); close(fd2[0]);
        for(i=0; i<4;)
        {
            read(fd1[0], &i, sizeof(int));
            printf("%d\n", 5-i);
            i++;
            write(fd2[1], &i, sizeof(int));
        }
        close(fd1[0]); close(fd2[1]);
        exit(0);
    }
    else
    {
        wait(&status);
        wait(&status);
        close(fd1[0]);
        close(fd1[1]);
        close(fd2[0]);
        close(fd2[1]);
        exit(0);
    }
    return 0;
}
```

ФИО: Хонакин, 21гр.

Задание 2.

```
// собиратель
int main()
{
    key_t keyshm;
    key_t keysem;
    int shmid;
    int semid;
    struct sembuf deblock = {1, 1, 0};
    char *shmaddr;
    char tree1, tree2, tree3, tree4;
    keyshm = ftok("shared", 23);
    keysem = ftok("sems", 46);
    semid = semget(keysem, 1, 0666|IPC_CREAT);
    shmid = shmget(keyshm, 4, 0666|IPC_CREAT);
    shmaddr = shmat(shmid, NULL, 0);
    scanf("%d %d %d %d", &tree1, &tree2, &tree3, &tree4);
    shmaddr[0]=tree1;  shmaddr[1] = tree2;
    shmaddr[2]=tree3;  shmaddr[3] = tree4;
    semop(semid, &deblock, 1);
    sleep(5);  // дождаться прогрева.
    shmdt(shmaddr);
    shmctl(shmid, IPC_RMID, NULL);
    semctl(semid, 0, IPC_RMID, NULL);
    return 0;
}
// вспомогательный;
int main()
{
    key_t keyshm;  struct sembuf block = {1, -1, 0};
    key_t keysem;  char *shmaddr;
    int shmid;  char tree1, tree2, tree3, tree4;
    int semid;
    keyshm = ftok("shared", 23);
    keysem = ftok("sems", 46);
    semid = semget(keysem, 1, 0666|IPC_CREAT);
    shmid = shmget(keyshm, 4, 0666|IPC_CREAT);
    shmaddr = shmat(shmid, NULL, 0);
    semop(semid, &block, 1);
    tree1 = shmaddr[0]; tree2 = shmaddr[1];
    tree3 = shmaddr[2]; tree4 = shmaddr[3];
    tree1 = 0.6 * (tree2 + tree3 + tree4);
    printf("%d|h", tree1);
    return 0;
}
```