«Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И.Ульянова (Ленина)»
СПбГЭТУ «ЛЭТИ»

Кафедра вычислительной техники

Отчет по по лабораторной работе №8 по дисциплине «Организация процессов и программирование в среде Linux»
Тема: «ВЗАИМОДЕЙСТВИЕ ПРОЦЕССОВ НА ОСНОВЕ СООБЩЕНИЙ»

Студент гр.8306 _____ Слепов А.Э.

Преподаватель _____ Разумовский Г.В.

Санкт-Петербург

2021

Цель работы

Знакомство с механизмом обмена сообщениями и системными вызовами приема и передачи сообщений.

Задание

1. Написать три программы, выполняющиеся параллельно и читающие один и тот же файл. Программа, которая хочет прочитать файл, должна передать другим программам запрос на разрешение операции и ожидать их ответа. Эти запросы программы передают через одну очередь сообщений. Ответы каждая программа должна принимать в свою локальную очередь. В запросе указываются: номер программы, которой посылается запрос, идентификатор очереди, куда надо передать ответ, и время посылки запроса. Начать выполнять операцию чтения файла программе разрешается только при условии получения ответов от двух других программ. Каждая программа перед отображением файла на экране должна вывести следующую информацию: номер программы и времена ответов, полученных от других программ. Программа, которая получила запрос от другой программы, должна реагировать следующим образом: если программа прочитала файл, то сразу передается ответ, который должен содержать номер отвечающей программы и время ответа; если файл не читался, то ответ передается только при условии, что время посылки запроса в сообщении меньше, чем время запроса на чтение у данной программы. Запросы, на которые ответы не были переданы, должны быть запомнены и после чтения файла обслужены.

2. Откомпилировать 3 программы и запустить их несколько раз на разных терминалах в различной последовательности

Порядок выполнения работы

В работе запускаются 3 программы, которые обмениваются сообщениями для синхронизации операции с чтением файла. Все программы посылают свои запросы на чтение в общую очередь сообщений. Затем программы проверяют

наличие в глобальной очереди запросов на чтение от других программ. Если в очереди имеются запросы с меньшим временем отправки, программа отправляет свое разрешение в личную очередь программы, которая отправила запрос раньше. Когда программа получает разрешения от двух других программ, она приступает к чтению файла. По окончании чтения файла, программа отправляет свое разрешение другим программам, которые ранее его запрашивали.

Результаты запуска программ в последовательности prog3 prog1 prog2 приведен на рисунке 1.

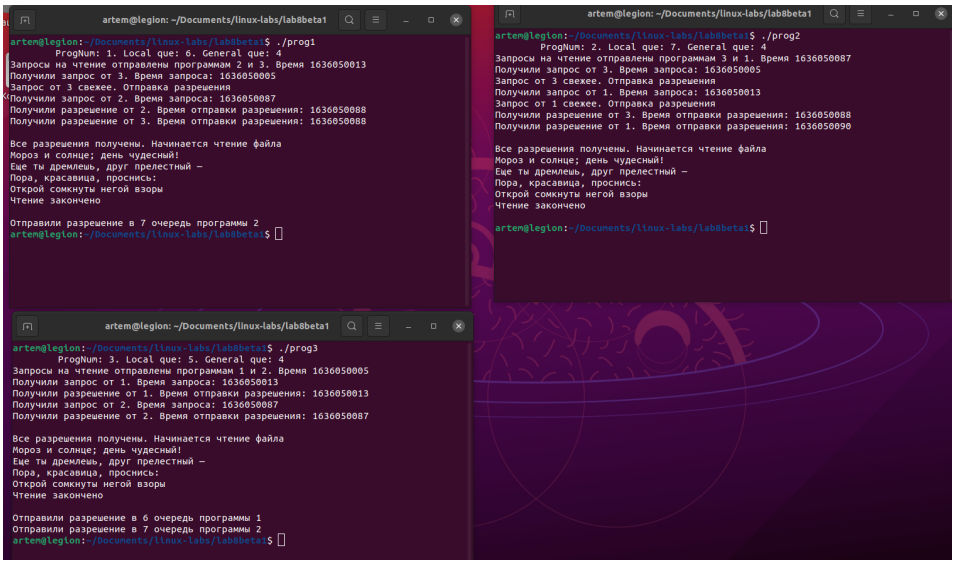

Рисунок 1. Результаты запуска в последовательности prog3 prog1 prog2

Результаты запуска программ в последовательности prog3 prog1 prog2 приведен на рисунке 2.

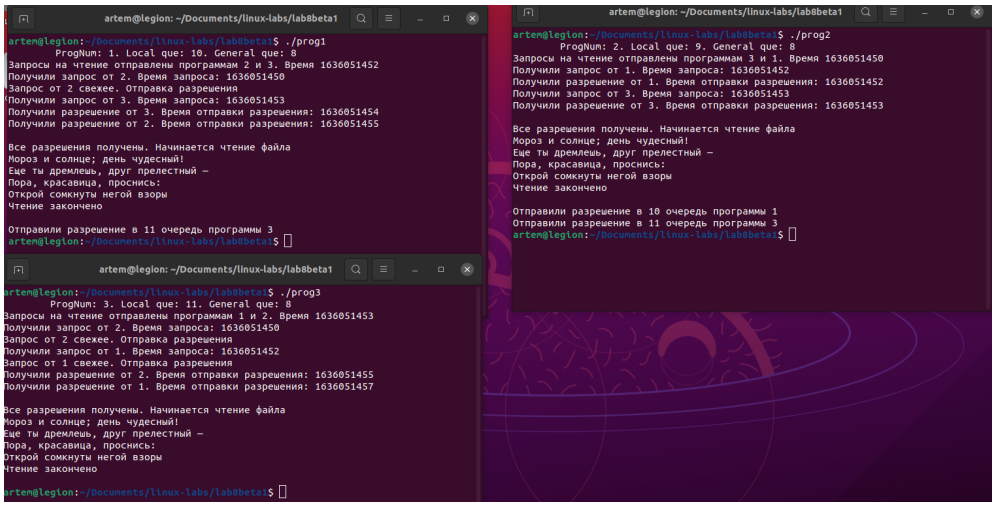

Рисунок 1. Результаты запуска в последовательности prog2 prog1 prog3

Выводы

В ходе работы были изучены механизмы обмена сообщениями и системными вызовами приема и передачи сообщений в операционной системе Ubuntu.

ПРИЛОЖЕНИЕ А
Текст программы №1

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <errno.h>
#include <time.h>
#include <unistd.h>

#define QUE_KEY 123

int general_que, local_que;
int progNum = 1;

int nextNum(int shift){
      if(shift == 1 &&  progNum == 1)     return 2;
      if(shift == 1 &&  progNum == 2) return 3;
      if(shift == 1 &&  progNum == 3) return 1;
      if(shift == 2 &&  progNum == 1)     return 3;
      if(shift == 2 &&  progNum == 2) return 1;
      if(shift == 2 &&  progNum == 3) return 2;
}

typedef struct MyMessage_T {
      long mtype;
      int sender;
      time_t time;
      int response_que;
      //char msg;
} MyMessage;

int main(){

      char queOwner = 1;

      general_que = msgget(QUE_KEY, 0606|IPC_CREAT|IPC_EXCL);
      if(general_que == -1){
            general_que =  msgget(QUE_KEY, 0606|IPC_CREAT);
            queOwner = 0;
      }
      local_que = msgget(IPC_PRIVATE, 0606|IPC_CREAT);

      printf("\tProgNum: %d. Local que: %d. General que: %d\n", progNum,
local_que, general_que);

      MyMessage myRequest, recieveGlobalMsg[2], recieveLocalMsg[2], myResponse;
      time_t myRequestTime = time(0);
      myRequest.time = myRequestTime;
      myRequest.mtype = nextNum(1);
      myRequest.response_que = local_que;
      myRequest.sender = progNum;
      msgsnd(general_que,&myRequest,sizeof(myRequest),0);
      myRequest.mtype = nextNum(2);
      msgsnd(general_que,&myRequest,sizeof(myRequest),0);

      printf("Запросы на чтение отправлены программам %d и %d. Время %ld\n",
nextNum(1), nextNum(2), myRequestTime);


    myResponse.sender = progNum;
```

```
      int readPermission = 0;
      int recieveLocalNum = 0, recieveGlobalNum = 0;
    while(recieveLocalNum < 2){
      if(msgrcv(general_que, &recieveGlobalMsg[recieveGlobalNum],
                  sizeof(recieveGlobalMsg[recieveGlobalNum]), progNum,
IPC_NOWAIT)!=-1){
            printf("Получили запрос от %d. Время запроса: %ld\n",
recieveGlobalMsg[recieveGlobalNum].sender,
recieveGlobalMsg[recieveGlobalNum].time);

            if(recieveGlobalMsg[recieveGlobalNum].time < myRequestTime){
                  recieveGlobalMsg[recieveGlobalNum].time = 0;
                  printf("Запрос от %d свежее. Отправка разрешения\n",
recieveGlobalMsg[recieveGlobalNum].sender);
                  myResponse.mtype = recieveGlobalMsg[recieveGlobalNum].sender;
                  myResponse.time = time(0);
                  msgsnd(recieveGlobalMsg[recieveGlobalNum].response_que,
&myResponse, sizeof(myResponse),0);
            }
            recieveGlobalNum+=1;
      }

      if(msgrcv(local_que, &recieveLocalMsg[recieveLocalNum],
sizeof(recieveLocalMsg[recieveLocalNum]), 0, IPC_NOWAIT)!=-1){
            printf("Получили разрешение от %d. Время отправки разрешения: %ld\
n", recieveLocalMsg[recieveLocalNum].sender,
recieveLocalMsg[recieveLocalNum].time);
            recieveLocalNum += 1;
      }
      //printf("recieveLocalNum: %d, recieveGlobalNum: %d\n", recieveLocalNum,
recieveGlobalNum);
      sleep(1);
    }


    printf("\nВсе разрешения получены. Начинается чтение файла\n");
    FILE *inputFile = fopen("input.txt","r");
    char str[32];
    while(fgets(str,32,inputFile))
      printf("%s",str);
    fclose(inputFile);
      printf("Чтение закончено\n\n");

      if(recieveGlobalMsg[0].time > 0){
            myResponse.mtype = recieveGlobalMsg[0].sender;
            myResponse.time = time(0);
            msgsnd(recieveGlobalMsg[0].response_que, &myResponse,
sizeof(myResponse),0);
            printf("Отправили разрешение в %d очередь программы %d\n",
recieveGlobalMsg[0].response_que, recieveGlobalMsg[0].sender);
      }

      if(recieveGlobalMsg[1].time > 0){
            myResponse.time = time(0);
            myResponse.mtype = recieveGlobalMsg[1].sender;
            msgsnd(recieveGlobalMsg[1].response_que, &myResponse,
sizeof(myResponse),0);
            printf("Отправили разрешение в %d очередь программы %d\n",
recieveGlobalMsg[1].response_que, recieveGlobalMsg[1].sender);
      }


    if(queOwner)
      msgctl(general_que,IPC_RMID,NULL);
```

```
    msgctl(local_que,IPC_RMID,NULL);
      return 0;
}
```

ПРИЛОЖЕНИЕ Б
Текст программы №2

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <errno.h>
#include <time.h>
#include <unistd.h>

#define QUE_KEY 123

int general_que, local_que;
int progNum = 2;

int nextNum(int shift){
      if(shift == 1 &&  progNum == 1)     return 2;
      if(shift == 1 &&  progNum == 2) return 3;
      if(shift == 1 &&  progNum == 3) return 1;
      if(shift == 2 &&  progNum == 1)     return 3;
      if(shift == 2 &&  progNum == 2) return 1;
      if(shift == 2 &&  progNum == 3) return 2;
}

typedef struct MyMessage_T {
      long mtype;
      int sender;
      time_t time;
      int response_que;
      //char msg;
} MyMessage;

int main(){

      char queOwner = 1;

      general_que = msgget(QUE_KEY, 0606|IPC_CREAT|IPC_EXCL);
      if(general_que == -1){
            general_que =  msgget(QUE_KEY, 0606|IPC_CREAT);
            queOwner = 0;
      }
      local_que = msgget(IPC_PRIVATE, 0606|IPC_CREAT);

      printf("\tProgNum: %d. Local que: %d. General que: %d\n", progNum,
local_que, general_que);

      MyMessage myRequest, recieveGlobalMsg[2], recieveLocalMsg[2], myResponse;
      time_t myRequestTime = time(0);
      myRequest.time = myRequestTime;
      myRequest.mtype = nextNum(1);
      myRequest.response_que = local_que;
      myRequest.sender = progNum;
      msgsnd(general_que,&myRequest,sizeof(myRequest),0);
      myRequest.mtype = nextNum(2);
      msgsnd(general_que,&myRequest,sizeof(myRequest),0);

      printf("Запросы на чтение отправлены программам %d и %d. Время %ld\n",
nextNum(1), nextNum(2), myRequestTime);


    myResponse.sender = progNum;
```

```
      int readPermission = 0;
      int recieveLocalNum = 0, recieveGlobalNum = 0;
    while(recieveLocalNum < 2){
      if(msgrcv(general_que, &recieveGlobalMsg[recieveGlobalNum],
                  sizeof(recieveGlobalMsg[recieveGlobalNum]), progNum,
IPC_NOWAIT)!=-1){
            printf("Получили запрос от %d. Время запроса: %ld\n",
recieveGlobalMsg[recieveGlobalNum].sender,
recieveGlobalMsg[recieveGlobalNum].time);

            if(recieveGlobalMsg[recieveGlobalNum].time < myRequestTime){
                  recieveGlobalMsg[recieveGlobalNum].time = 0;
                  printf("Запрос от %d свежее. Отправка разрешения\n",
recieveGlobalMsg[recieveGlobalNum].sender);
                  myResponse.mtype = recieveGlobalMsg[recieveGlobalNum].sender;
                  myResponse.time = time(0);
                  msgsnd(recieveGlobalMsg[recieveGlobalNum].response_que,
&myResponse, sizeof(myResponse),0);
            }
            recieveGlobalNum+=1;
      }

      if(msgrcv(local_que, &recieveLocalMsg[recieveLocalNum],
sizeof(recieveLocalMsg[recieveLocalNum]), 0, IPC_NOWAIT)!=-1){
            printf("Получили разрешение от %d. Время отправки разрешения: %ld\
n", recieveLocalMsg[recieveLocalNum].sender,
recieveLocalMsg[recieveLocalNum].time);
            recieveLocalNum += 1;
      }
      //printf("recieveLocalNum: %d, recieveGlobalNum: %d\n", recieveLocalNum,
recieveGlobalNum);
      sleep(1);
    }


    printf("\nВсе разрешения получены. Начинается чтение файла\n");
    FILE *inputFile = fopen("input.txt","r");
    char str[32];
    while(fgets(str,32,inputFile))
      printf("%s",str);
    fclose(inputFile);
      printf("Чтение закончено\n\n");

      if(recieveGlobalMsg[0].time > 0){
            myResponse.mtype = recieveGlobalMsg[0].sender;
            myResponse.time = time(0);
            msgsnd(recieveGlobalMsg[0].response_que, &myResponse,
sizeof(myResponse),0);
            printf("Отправили разрешение в %d очередь программы %d\n",
recieveGlobalMsg[0].response_que, recieveGlobalMsg[0].sender);
      }

      if(recieveGlobalMsg[1].time > 0){
            myResponse.time = time(0);
            myResponse.mtype = recieveGlobalMsg[1].sender;
            msgsnd(recieveGlobalMsg[1].response_que, &myResponse,
sizeof(myResponse),0);
            printf("Отправили разрешение в %d очередь программы %d\n",
recieveGlobalMsg[1].response_que, recieveGlobalMsg[1].sender);
      }


    if(queOwner)
      msgctl(general_que,IPC_RMID,NULL);
```

```
    msgctl(local_que,IPC_RMID,NULL);
      return 0;
}
```

ПРИЛОЖЕНИЕ В
Текст программы №3

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <errno.h>
#include <time.h>
#include <unistd.h>

#define QUE_KEY 123

int general_que, local_que;
int progNum = 3;

int nextNum(int shift){
      if(shift == 1 &&  progNum == 1)     return 2;
      if(shift == 1 &&  progNum == 2) return 3;
      if(shift == 1 &&  progNum == 3) return 1;
      if(shift == 2 &&  progNum == 1)     return 3;
      if(shift == 2 &&  progNum == 2) return 1;
      if(shift == 2 &&  progNum == 3) return 2;
}

typedef struct MyMessage_T {
      long mtype;
      int sender;
      time_t time;
      int response_que;
      //char msg;
} MyMessage;

int main(){

      char queOwner = 1;

      general_que = msgget(QUE_KEY, 0606|IPC_CREAT|IPC_EXCL);
      if(general_que == -1){
            general_que =  msgget(QUE_KEY, 0606|IPC_CREAT);
            queOwner = 0;
      }
      local_que = msgget(IPC_PRIVATE, 0606|IPC_CREAT);

      printf("\tProgNum: %d. Local que: %d. General que: %d\n", progNum,
local_que, general_que);

      MyMessage myRequest, recieveGlobalMsg[2], recieveLocalMsg[2], myResponse;
      time_t myRequestTime = time(0);
      myRequest.time = myRequestTime;
      myRequest.mtype = nextNum(1);
      myRequest.response_que = local_que;
      myRequest.sender = progNum;
      msgsnd(general_que,&myRequest,sizeof(myRequest),0);
      myRequest.mtype = nextNum(2);
      msgsnd(general_que,&myRequest,sizeof(myRequest),0);

      printf("Запросы на чтение отправлены программам %d и %d. Время %ld\n",
nextNum(1), nextNum(2), myRequestTime);


    myResponse.sender = progNum;
```

```
      int readPermission = 0;
      int recieveLocalNum = 0, recieveGlobalNum = 0;
    while(recieveLocalNum < 2){
      if(msgrcv(general_que, &recieveGlobalMsg[recieveGlobalNum],
                  sizeof(recieveGlobalMsg[recieveGlobalNum]), progNum,
IPC_NOWAIT)!=-1){
            printf("Получили запрос от %d. Время запроса: %ld\n",
recieveGlobalMsg[recieveGlobalNum].sender,
recieveGlobalMsg[recieveGlobalNum].time);

            if(recieveGlobalMsg[recieveGlobalNum].time < myRequestTime){
                  recieveGlobalMsg[recieveGlobalNum].time = 0;
                  printf("Запрос от %d свежее. Отправка разрешения\n",
recieveGlobalMsg[recieveGlobalNum].sender);
                  myResponse.mtype = recieveGlobalMsg[recieveGlobalNum].sender;
                  myResponse.time = time(0);
                  msgsnd(recieveGlobalMsg[recieveGlobalNum].response_que,
&myResponse, sizeof(myResponse),0);
            }
            recieveGlobalNum+=1;
      }

      if(msgrcv(local_que, &recieveLocalMsg[recieveLocalNum],
sizeof(recieveLocalMsg[recieveLocalNum]), 0, IPC_NOWAIT)!=-1){
            printf("Получили разрешение от %d. Время отправки разрешения: %ld\
n", recieveLocalMsg[recieveLocalNum].sender,
recieveLocalMsg[recieveLocalNum].time);
            recieveLocalNum += 1;
      }
      //printf("recieveLocalNum: %d, recieveGlobalNum: %d\n", recieveLocalNum,
recieveGlobalNum);
      sleep(1);
    }


    printf("\nВсе разрешения получены. Начинается чтение файла\n");
    FILE *inputFile = fopen("input.txt","r");
    char str[32];
    while(fgets(str,32,inputFile))
      printf("%s",str);
    fclose(inputFile);
      printf("Чтение закончено\n\n");

      if(recieveGlobalMsg[0].time > 0){
            myResponse.mtype = recieveGlobalMsg[0].sender;
            myResponse.time = time(0);
            msgsnd(recieveGlobalMsg[0].response_que, &myResponse,
sizeof(myResponse),0);
            printf("Отправили разрешение в %d очередь программы %d\n",
recieveGlobalMsg[0].response_que, recieveGlobalMsg[0].sender);
      }

      if(recieveGlobalMsg[1].time > 0){
            myResponse.time = time(0);
            myResponse.mtype = recieveGlobalMsg[1].sender;
            msgsnd(recieveGlobalMsg[1].response_que, &myResponse,
sizeof(myResponse),0);
            printf("Отправили разрешение в %d очередь программы %d\n",
recieveGlobalMsg[1].response_que, recieveGlobalMsg[1].sender);
      }


    if(queOwner)
      msgctl(general_que,IPC_RMID,NULL);
```

```
    msgctl(local_que,IPC_RMID,NULL);
      return 0;
}
```

ПРИЛОЖЕНИЕ Г
Текст скрипта для запуска всех программ

```
#!/bin/sh
rm 1.txt
rm 2.txt
gcc proc1.c -o proc1
gcc proc2.c -o proc2
gcc lab7.c
./a.out input.txt 1.txt 2.txt
```