Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №6
По дисциплине: «Операционные системы и системное программирование»
Тема: «средства межпроцессного взаимодействия»

Подготовил:
Студент 2 курса
Группы ПО-3(2)
Огиевич Е.А.
Проверила:
Давидюк Ю.И.

Брест, 2020

Лабораторная работа №6
Вариант 20

Цель: изучить средства межпроцессного взаимодействия.

Задание:

Очереди сообщений: родитель передаёт потомку 3 строки, потомок возвращает эти строки обратно, но в каждой строке символы отсортированы по убыванию.

Текст программы:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <mqueue.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/stat.h>
#define MQNAME "/m"

/*********************************************/
/*****ПРИНУДИТЕЛЬНО ПРОГРАММУ НЕ ЗАВЕРШАТЬ*****/
/*********************************************/

int main (void)
{
int t1 = getpid();
printf("PID процесса: %d\n", t1); // выводим на экран ID текущего (родительского процесса)

char f[1024]; // будем тут хранить поток
int reads;
if((reads = read(0, f, sizeof(f)) < 0)) // из стандартного потока читаем данные и записываем в чаровый массив
{
printf("ERROR");
}

// Создаём объект, чтобы настроить параметры очереди и при открытии ссылать на него
struct mq_attr mqAttr;
int rc; // переменная чтобы хранить статус отправки
printf("Началась отпавка сообщения.\n");
//Устанавливаем аттрибуты очереди
mqAttr.mq_maxmsg = 10;
mqAttr.mq_msgsize = 1024;
//Открываем очередь
mqd_t sndHndl;
```

```
// Открываем очередь, если не удачно - ошибка и выход
sndHndl = mq_open(MQNAME, O_RDWR|O_CREAT, S_IWUSR|S_IRUSR|S_IRGRP|S_IWGRP|
S_IROTH|S_IWOTH, &mqAttr);
if (sndHndl < 0)
{
printf ("\t\tError %d (%s) mq_open for send.\n",
errno, strerror (errno));
exit (0);
}
printf ("Очередь успешно открыта\n");

//посылаем сообщение в очередь
rc = mq_send(sndHndl, f, sizeof (f), 1);
if (rc < 0)
{
printf ("\t\tError %d (%s) mq_send.\n",
errno, strerror (errno));
exit (0);
}
printf("Сообщение успешно отправлено\n");

int pid_st; // возвращаем сюда результат форка
char f1[1024]; // сюда будем принимать наши слова для обработки
mqd_t rcHndl;
int rc1;

// Создаём новый процесс
if((pid_st = fork()) == -1)
{
printf("ERROR: fork returns -1");
}
else if (pid_st == 0) // Если 0, значит мы в ребенке
{
int t2 = getpid();
printf("Процесс %d успешно запущен\n", t2);
// В другом процессе опять открываем очередь и берем оттуда данные
rcHndl = mq_open(MQNAME, O_RDWR);
if (rcHndl < 0)
{
printf ("\t\tError %d (%s) mq_open for receive.\n",
errno, strerror (errno));
exit (-1);
}
printf ("Очередь успешно открыта\n");
// Принимаем сообщение из очереди
rc1 = mq_receive(rcHndl, f1, sizeof (f1), NULL);
if (rc1 < 0)
{
printf ("\t\tError %d (%s) mq_receive.\n",
errno, strerror (errno));
exit (-1);
```

```
}
printf ("Сообщение успешно принято:\n %s\n", f1);
// Начинаем обработку сообщения
// Сразу всё пихаем в двумерный массив, 1 слово - 1 строка, чтобы можно было слова отдель обработать
char buf[20][20]; int lst_index[20];
int j = 0, i = 0, q = 0;
while(f[q] != '\0')
{
if(f1[q] == '\n') // Если находим этот символ строка переходит дальше и записываем пока снова не встретим
{
lst_index[i] = j; // Нужно знать последние индексы в строках, потому что С не С++ и если не
j = 0; q++; i++; // делать как в след. цикле то будет выводить мусор
}
else
{
buf[i][j] = f1[q];
j++; q++;
}
}

// Промежуточный результат выводим на экран
// Чтобы убедится что нормально записало
for(int p = 0; p < i; p++)
{
for(int j = 0; j < lst_index[p]; j++)
{
printf("%c", buf[p][j]);
}
printf("\n");
}


// Алгоритм сортировки, сравнивает каждый символ с каждым в строке, как будто бы массив одномерный
// и меняем местами буковки
for (int q = 0; q < i; q++)
{
for (int i = 0; i < lst_index[q] - 1; i++)
{
for (int j = lst_index[q] - 1; j>i; j--)
{
if(buf[q][j] > buf[q][j-1])
{
int tmp = buf[q][j];
buf[q][j] = buf[q][j-1];
buf[q][j-1] = tmp;
}
}
```

```
}
}
// Опять всё сводим к одномерному массиву, чтобы вернуть просто строчку с
отсортированными словами
char buf_o[1024]; int current_j = 0;
for(int p = 0; p < i; p++)
{
for(int j = 0; j < lst_index[p]; j++)
{
buf_o[current_j] = buf[p][j]; current_j++;
}
current_j++;
buf_o[current_j] = ' '; // если 1 слово закончилось пихаем пробел
current_j++;
}

// опять открываем очередь
int sndHndl_back;
sndHndl_back = mq_open(MQNAME, O_RDWR);
if (sndHndl_back < 0)
{
printf ("\t\tError %d (%s) mq_open for receive.\n",
errno, strerror (errno));
exit (-1);
}
printf ("Очередь успешно открыта\n");
// Отправляем в очередь наш массив
int snd_back;
snd_back = mq_send(sndHndl_back, buf_o, sizeof (buf_o), 1);
if (snd_back < 0)
{
printf ("\t\tError %d (%s) mq_send.\n",
errno, strerror (errno));
exit (-1);
}
printf("Сообщение успешно отправлено\n");
int t5 = getpid();
printf("Процесс %d уничтожен.\n", t5);
exit(0);
} else sleep(2); // <- как выглядит 6 часов жизни

int t3 = getpid();
printf("ID текущего процесса: %d\n", t3); // удостоверились что вернулись к
родительскому процессу
sleep(3);
// Опять открываем очередь
int sndHndl_back2;
sndHndl_back2 = mq_open(MQNAME, O_RDWR);
if (sndHndl_back2 < 0)
{
printf ("\t\tError %d (%s) mq_open for receive.\n",
```

```
errno, strerror (errno));
exit (-1);
}
printf ("Очередь успешно открыта.\n");
// Принимаем сообщение из очереди
int rc2; char result[1024];
rc2 = mq_receive(sndHndl_back2, result, sizeof (result), NULL);
if (rc2 < 0)
{
printf ("\t\tError %d (%s) mq_receive.\n",
errno, strerror (errno));
exit (-1);
}
printf("Сообщение успешно получено.\n");

// Выводим на экран результат
printf("Результат: \n");
for (int i = 0; i < strlen(result); i++)
{
printf("%c",result[i]);
}

//printf("checker: %c", result[11]);
mq_unlink(MQNAME);
return 0;
}
```

Результат выполнения:

```
natefoust@natefoust-X550LC:~/OS/lab6$ ls | ./a.out
PID процесса: 6844
Началась отпавка сообщения.
Очередь успешно открыта
Сообщение успешно отправлено
Процесс 6845 успешно запущен
Очередь успешно открыта
Сообщение успешно принято:
 a.out
file
lab6.c
m

a.out
file
lab6.c
m
Очередь успешно открыта
Сообщение успешно отправлено
Процесс 6845 уничтожен.
ID текущего процесса: 6844
Очередь успешно открыта.
Сообщение успешно получено.
Результат:
utoa. life◆ lcba6.◆ mnatefoust@natefoust-X550LC:~/OS/lab6$
```

Вывод: изучил средства межпроцессного взаимодействия, написал программу по передаче данных между процессами с помощью очередей сообщений.