**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Операционные системы»

**Курсовой проект по курсу «Операционные системы»**  
Вариант на «удовлетворительно»

Студент: Тимофеев А.В.

Группа: 80-207

Преподаватель: Миронов Е. С.

Дата:

Оценка:

**Постановка задачи**

Цель курсового проекта

1. Приобретение практических навыков в использовании знаний, полученных в течении курса

2. Проведение исследования в выбранной предметной области

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 на «удовлетворительно»

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

С. Отправка строк должна производится построчно. Программа C печатает в стандартый вывод,

полученную строку от программы A. После получения программа C отправляет программе А

сообщение о том, что строка получена. До тех пор пока программа А не примет «сообщение о

получение строки» от программы С, она не может отправялять следующую строку программе С.Программа B пишет в стандартный вывод количество отправленных символов программой А и

количество принятых символов программой С. Данную информацию программа B получает от

программ A и C соответственно.

Способ организация межпроцессорного взаимодействия выбирает студент

**Алгоритм решения**

Создаем три программы А В и С. А будет главная программа, которая будет передавать через отображаемый в памяти файл строки программе В и числа программе С. Создадим отображаемый файл для связи В и С программ. Создадим два дочерних процесса для программы В и С. В них передадим с помощью execv управление программам В и С, также передадим туда pid родителя и названия отображаемых в памяти файлов.

Программа А считывает строки из стандартного потока ввода, считает количество символов в них и передает строку в B а количество символов в C.

Программа В получает название каналов сообщения, подключается к ним, далее получает строки, считать количество символов в строке и отправляет его в программу С.

Программа С получает название каналов сообщения, подключается к ним, далее получает количество символов в строке из программы А, выводит эту информацию, далее получает количество символов в строке из программы В, выводит эту информацию.

**Листинг программы**

**mainA.c**

#include "unistd.h"

#include "stdio.h"

#include "stdlib.h"

#include "sys/mman.h"

#include "string.h"

#include "signal.h"

#include "fcntl.h"

#include <sys/types.h>

#define check(VALUE, MSG, BADVAL) if (VALUE == BADVAL) { perror(MSG); exit(1); }

int main() {

size\_t pagesize = sysconf(\_SC\_PAGESIZE);

int pid = getpid();

char mmfilename1AC[] = "mmfile1AC";

char mmfilename2AB[] = "mmfile2AB";

char mmfilename3BC[] = "mmfile3BC";

int mmf1AC = open(mmfilename1AC, O\_RDWR | O\_CREAT, S\_IRWXU);

check(mmf1AC, "open 1 error", 0)

int mmf2AB = open(mmfilename2AB, O\_RDWR | O\_CREAT, S\_IRWXU);

check(mmf2AB, "open 2 error", 0)

int mmf3BC = open(mmfilename3BC, O\_RDWR | O\_CREAT, S\_IRWXU);

check(mmf3BC, "open 3 error", 0)

size\_t i1 = 0, i2 = 0;

ftruncate(mmf1AC, pagesize);

ftruncate(mmf2AB, pagesize);

ftruncate(mmf3BC, pagesize);

int id1 = fork();// первый дочерний процесс

if (id1 == -1) {

perror("fork 1 error");

return -1;

}else if (id1 > 0){// родительский процесс

int id2 = fork();// второй дочерний процесс

if (id2 == -1) {

perror("fork 2 error");

return -1;

}else if (id2 > 0){// родительский процесс

char \*fmap1 = (char \*)mmap(NULL, pagesize, PROT\_WRITE | PROT\_READ, MAP\_SHARED, mmf1AC, 0);

check(fmap1, "mmap 1 error", MAP\_FAILED)

char \*fmap2 = (char \*)mmap(NULL, pagesize, PROT\_WRITE | PROT\_READ, MAP\_SHARED, mmf2AB, 0);

check(fmap2, "mmap 2 error", MAP\_FAILED)

sigset\_t set;

check(sigemptyset(&set), "sigemptyset error", -1)

check(sigaddset(&set, SIGUSR1), "sigaddset error", -1)

check(sigprocmask(SIG\_BLOCK, &set, NULL), "sigprocmask error", -1)

int sig;

char c;

int n = 0;

while (scanf("%c", &c) > 0){// заполняем pipes

if (c != '\n'){

++n;

fmap2[i2] = c;

if (++i2 == pagesize) {

i2 = 0;

check(kill(id2, SIGUSR1), "send signal to B.out 2 error", -1)

check(sigwait(&set, &sig), "sigwait error", -1)

}

}else{

fmap2[i2] = c;

i2 = 0;

check(kill(id2, SIGUSR1), "send signal to B.out 2 error", -1)

check(sigwait(&set, &sig), "sigwait error", -1)

sprintf(fmap1,"%d %c",n,'\n');

i1 = 0;

n = 0;

check(kill(id1, SIGUSR1), "send signal to C.out 1 error", -1)

check(sigwait(&set, &sig), "sigwait error", -1)

}

}

c = '\0';

fmap1[0] = c;

fmap2[0] = c;

check(kill(id1, SIGUSR1), "send signal to C.out 1 error", -1)

check(kill(id2, SIGUSR1), "send signal to B.out 2 error", -1)

check(munmap(fmap1, pagesize), "munmap error", -1);

check(munmap(fmap2, pagesize), "munmap error", -1);

close(mmf1AC);

close(mmf2AB);

close(mmf3BC);

}else{// второй дочерний процесс

close(mmf1AC);

char spid[10];

snprintf(spid, 10, "%d", pid);

char\* const args[] = {"B.out", mmfilename2AB, spid, mmfilename3BC, (char \*)NULL};

check(execv("B.out", args), "execv B.out 2 error", -1)

}

}else{// первый дочерний процесс

close(mmf2AB);

char spid[10];

snprintf(spid, 10, "%d", pid);

char\* const args[] = {"C.out", mmfilename1AC, spid, mmfilename3BC, (char \*)NULL};

check(execv("C.out", args), "execv C.out 1 error", -1)

}

return 0;

}

**B.c**

#include "unistd.h"

#include "stdio.h"

#include "stdlib.h"

#include "sys/mman.h"

#include "string.h"

#include "signal.h"

#include "fcntl.h"

#define check(VALUE, MSG, BADVAL) if (VALUE == BADVAL) { perror(MSG); exit(-1); }

int main(int agrc, char const \*argv[]) {

size\_t pagesize = sysconf(\_SC\_PAGESIZE);

char c = '\0';

int n = 0;

char\* str = (char \*)malloc(sizeof(char) \* pagesize);

check(str, "malloc error", NULL)

int mfileAB = open(argv[1], O\_RDWR);

check(mfileAB, "open error", -1)

int pid = atoi(argv[2]);

char\* fmapAB = (char \*)mmap(NULL, pagesize, PROT\_READ, MAP\_SHARED, mfileAB, 0);

check(fmapAB, "mmap error", MAP\_FAILED)

size\_t i = 0;

int mfileBC = open(argv[3], O\_RDWR);

check(mfileBC, "open error", -1)

char\* fmapBC = (char \*)mmap(NULL, pagesize, PROT\_WRITE | PROT\_READ, MAP\_SHARED, mfileBC, 0);

check(fmapBC, "mmap error", MAP\_FAILED)

sigset\_t set;

check(sigemptyset(&set), "sigemptyset error", -1)

check(sigaddset(&set, SIGUSR1), "sigaddset error", -1)

check(sigprocmask(SIG\_BLOCK, &set, NULL), "sigprocmask error", -1)

int sig;

for(;;) {

check(sigwait(&set, &sig), "sigwait error", -1);

printf("B:\t");

for (i = 0; i < pagesize; ++i) {

c = fmapAB[i];

if (c != '\n' && c != '\0') {

str[i] = c;

++n;

} else if (c == '\0') {

break;

} else {

for (int i = 0; i < n; i++) {

printf("%c", str[i]);

}

printf("%c", c);

sprintf(fmapBC,"%d %c",n,'\n');

n = 0;

break;

}

}

if (c == '\0') {

break;

} else {

check(kill(pid, SIGUSR1), "send signal to parent error", -1)

}

}

free(str);

check(munmap(fmapAB, pagesize), "munmap error", -1)

check(munmap(fmapBC, pagesize), "munmap error", -1)

close(mfileAB);

close(mfileBC);

}

**C.c**

#include "unistd.h"

#include "stdio.h"

#include "stdlib.h"

#include "sys/mman.h"

#include "string.h"

#include "signal.h"

#include "fcntl.h"

#define check(VALUE, MSG, BADVAL) if (VALUE == BADVAL) { perror(MSG); exit(-1); }

int main(int agrc, char const \*argv[]) {

size\_t pagesize = sysconf(\_SC\_PAGESIZE);

char c = '\0';

char c1 = '\0';

char\* str = (char \*)malloc(sizeof(char) \* pagesize);

check(str, "malloc error", NULL)

int mfileAC = open(argv[1], O\_RDWR);

check(mfileAC, "open error", -1)

int pid = atoi(argv[2]);

char\* fmapAC = (char \*)mmap(NULL, pagesize, PROT\_READ, MAP\_SHARED, mfileAC, 0);

check(fmapAC, "mmap error", MAP\_FAILED)

size\_t i = 0;

int mfileBC = open(argv[3], O\_RDWR);

check(mfileBC, "open error", -1)

char\* fmapBC = (char \*)mmap(NULL, pagesize, PROT\_WRITE | PROT\_READ, MAP\_SHARED, mfileBC, 0);

check(fmapBC, "mmap error", MAP\_FAILED)

sigset\_t set;

check(sigemptyset(&set), "sigemptyset error", -1)

check(sigaddset(&set, SIGUSR1), "sigaddset error", -1)

check(sigprocmask(SIG\_BLOCK, &set, NULL), "sigprocmask error", -1)

int sig;

for(;;) {

check(sigwait(&set, &sig), "sigwait error", -1);

printf("C:\t");

for (i = 0; i < pagesize; ++i) {

c = fmapAC[i];

if (c != '\n' && c != '\0') {

printf("%c", c);

} else if (c == '\0') {

break;

} else {

printf("\t");

break;

}

}

for (i = 0; i < pagesize; ++i) {

c = fmapBC[i];

if (c != '\n' && c != '\0') {

printf("%c", c);

} else if (c == '\0') {

break;

} else {

printf("%c", c);

break;

}

}

if (c == '\0' && c1 == '\0') {

break;

} else {

check(kill(pid, SIGUSR1), "send signal to parent error", -1)

}

}

free(str);

check(munmap(fmapAC, pagesize), "munmap error", -1)

check(munmap(fmapBC, pagesize), "munmap error", -1)

close(mfileAC);

close(mfileBC);

}

**Тесты и протокол исполнения**

dude@dude:~/Рабочий стол/OC/OC\_KP$ ./A.out

qwerty

B: qwerty

C: 6 6

qwrqwrgsvjdfjokgk((\*%)(Y5049

B: qwrqwrgsvjdfjokgk((\*%)(Y5049

C: 28 28

podrgmjikdmrgmdmrmgdrmgdmrggg

B: podrgmjikdmrgmdmrmgdrmgdmrggg

C: 29 29

**Вывод**:

Во время выполнения курсового проекта я упрочил свои навыки создания связи в виде отображаемых в память файлов, также я повторил создание процессов и управление ими, еще я вспомнил сигналы. Объедение эти возможности OC Unix и возможности языка Си, получился хороший правильно работающий курсовой проект.

**Список литературы**

1. Таненбаум Э., Бос Х. Современные операционные системы. – 4-е изд. – СПб.: Издательский дом «Питер», 2018.
2. Файлы, отображаемые в память – URL : https://habr.com/ru/post/55716/