Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №4 по курсу

«Операционные системы»

Обмен данными

Студент: [ Нелюбин В.С ]

Группа: М8О–206Б–19

Вариант: 10

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2020.

Постановка задачи

**Цель работы**

Приобретение практических навыков в:

* Освоение принципов работы с файловыми системами
* Обеспечение обмена данных между процессами посредством технологии «File mapping»

## Задание

Составить и отладить программу на языке Си, осуществляющую работу с процессами и взаимодействие между ними в одной из двух операционных систем. В результате работы программа (основной процесс) должен создать для решение задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или через отображаемые файлы (memory-mapped files).

Задание тоже, что и в лабораторной работе №2: Подпроцесс проверяет числа на простоту, родительский процесс выводит их на экран. Но обмен данными теперь происходит с помощью отображения файла.

**Общие сведения о программе**

Программа компилируется из файла whatt.c и whato.c (компилируются как c.out и b.out соответственно). Также используется заголовочные файлы: stdio.h stdlib.h unistd.h sys/types.h errno.h sys/mman.h sys/stat.h fcntl.h signal.h. Основные команды программы:

1. **fork** – Создаёт копию текущего процесса. Возвращает 0 в новом процессе, отрицательное значение при ошибке, и положительное значение в изначальном процессе. Часто используется с exec() или схожими функциями.
2. **execvp** – заменяет образ текущего процесса. Получает на вход имя исполнимого файла, и список аргументов для сказанного файла. Является частью семьи функций execl\*.
3. **mmap()** – отображение файла в память. Позволяет получать данные из файла, как из массива char\*. Все изменения мгновенны, даже если файл открыт другой программой.

Обе программы отображают указанный файл в память. Оба процесса считывают одинаковое количество данных. Обработав данные, подпроцесс ставит флаг в файле. Родительский процесс ждёт, а затем считывает флаг. В зависимости от флага, программа либо завершается, либо выводит обработанное число на экран. В конце работы родительский процесс возвращает данные файла в исходное состояние.

**Общий метод и алгоритм решения**.

Для реализации поставленной задачи необходимо:

1. Изучить принципы работы mmap().
2. Придумать, как использовать mmap() для решения данной задачи.
3. Переделать программу для работы с отображаемыми файлами.
4. Найти средства измерения времени и количества потоков программы

**Основные файлы программы**

**whatt.c:**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <errno.h>

#include <sys/mman.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <signal.h>

int main(){

int num=255;

char\*inpt2=malloc(sizeof(char)\*num);

char\*inpt;

fgets (inpt2, num, stdin);

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

if(inpt2[i]=='\n'){

num=i;

inpt=malloc(sizeof(char)\*num);

for(int ee=0;ee<num;ee++){

inpt[ee]=inpt2[ee];

}

free(inpt2);

break;

}

}

int fd = open(inpt, O\_RDWR);

if (fd == -1){printf("error opening file\n");}

char ch;

size\_t sz;

sz = lseek(fd, 0, SEEK\_END);

void\*mapped=mmap(NULL, sz, PROT\_READ|PROT\_WRITE,MAP\_SHARED, fd,0);

close(fd);

if(mapped==NULL){printf("bad mmap\n");return -1;}

char\*flhr=(char\*)mapped;

char\*backup=malloc(sizeof(char)\*sz);

if(backup==NULL){return -1;}

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

backup[i]=flhr[i];

}

pid\_t pid;

int ci,cim=0;

char\*szarg;

for(ci=1;ci<sz;ci\*=10){

cim++;

}

szarg=malloc(sizeof(char)\*(1+cim));

szarg[cim]='\0';

cim=0;

for(ci=ci/10;ci;ci/=10){

szarg[cim]='0'+(sz/ci)%10;

cim++;

}

pid=fork();

if(pid==0){

char\*args[]={"./b.out",inpt,szarg,NULL};

execvp(args[0],args);

\_exit (EXIT\_FAILURE);

}

else if(pid<0){

printf("something went wrong with fork()\n");

}

else{

ci=0,cim=0;

while(ci<sz){

while((flhr[ci]!='\n')&&(flhr[ci]!='Y')&&(flhr[ci]!='N')){

cim=cim\*10+flhr[ci]-'0';

ci++;

}

while(flhr[ci]=='\n');

if(flhr[ci]=='Y'){

printf("%d\n",cim);

cim=0;

ci++;

}

if(flhr[ci]=='N'){

break;

}

}

printf("prime number\n");

kill(pid, SIGKILL);

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

flhr[i]=backup[i];

}

free(backup);

munmap(mapped,sz);

}

return 0;

}

**whato.c**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <errno.h>

#include <sys/mman.h>

#include <sys/stat.h>

#include <fcntl.h>

int w(int sb){

for(int i=2;i\*i<sb;i++){

if((sb%i)==0){

return 0;

}

}

return 1;

}

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

if( argc != 3 ) {

printf("bad command line arguments :(\n");

exit(-1);

}

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

if (fd == -1){printf("error opening file\n");}

char ch;

size\_t sz=0;

for(int ee=0;argv[2][ee]!='\0';ee++){

sz=sz\*10+argv[2][ee]-'0';

}

char\*file=mmap(NULL, sz, PROT\_READ|PROT\_WRITE,MAP\_SHARED, fd, 0);

int wrtnm=0;

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

if(file[i]=='\n'){

if(w(wrtnm)){

file[i]='N';

}

else{

file[i]='Y';

}

i++;

wrtnm=0;

}

wrtnm=wrtnm\*10+file[i]-'0';

return 0;

}

**Пример работы**

aaaaa@TRASHBox:~/Desktop/OC/laba4$ ./c.out

input.txt

33

33

333

prime number

aaaaa@TRASHBox:~/Desktop/OC/laba4$ cat input.txt

33

33

333

3

3

aaaaa@TRASHBox:~/Desktop/OC/laba4**$**

**Вывод**

Отображение файлов, или кусков файлов, в память – крайне полезная вещь. Возможно, это самый простой способ обмена данными между процессами.