Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №4 по курсу

«Операционные системы»

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

Студент: Аминов Степан Сеегеевич
Группа: М80 – 308Б
Вариант: 3
Преподаватель: Миронов Евгений Сергеевич
Дата:
Оценка:
Подпись: _____

Москва, 2021

1 Постановка задачи

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).

Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса пишет имя файла, которое будет передано при создании дочернего процесса. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс передает команды пользователя через pipe1, который связан с стандартным входным потоком дочернего процесса. Дочерний процесс принеобходимости передает данные в родительский процесс через pipe2. Результаты своей работы дочерний процесс пишет в созданный им файл. Допускается просто открыть файл и писать туда, не перенаправляя стандартный поток вывода.

3 вариант) Пользователь вводит команды вида: «число число число<endline>». Далее эти числа передаются от родительского процесса в дочерний. Дочерний процесс производит деление первого числа, на последующие, а результат выводит в файл. Если происходит деление на 0, то тогда дочерний и родительский процесс завершают свою работу. Проверка деления на 0 должна осуществляться на стороне дочернего процесса. Числа имеют тип int.

2 Общий метод и алгоритм решения

При запуске программы пользователю предлагается ввести имя файла. В этот файл будет записываться вывод соответствующих процессов. Если пользователь ввёл имя несуществующего файла, он будет создан.

После запуска программы создаётся дочерний процесс. Родительский процесс считывает числа с консольного ввода. Передача чисел дочернему процессу осуществляется посредством их копирования в отображенный файл. Дочерний процесс производит необходимые вычисления и выводит результат в созданный файл.

3 Исходный код

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/types.h>
#include <errno.h>
#include <string.h>

int main(void)
{
   int numbers[3];
   char *ptr;
   char filename[100];
   char *maped = "MapedFile";

   scanf("%s", filename);
   for(int i = 0; i < 3 ;i++) {
      scanf("%d", &numbers[i]);
   }

   unsigned int fd = open(maped, O_RDWR | O_CREAT  , S_IWRITE | S_IREAD);
   if(fd == -1){
      printf("Error file descriptor \n");
      exit(1);
   }
   ptr = mmap(NULL, 3*sizeof(int), PROT_WRITE, MAP_SHARED, fd, 0);
   if(ptr == MAP_FAILED)
   {
      printf("Map failed in write process: %s\n", strerror(errno));
      exit(1);
   }
   memcpy( ptr,numbers, 3*sizeof(int ));

   execl("child"," ",filename, NULL);

   close(fd);

   return 0;
}
```

child.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/types.h>
#include <errno.h>
#include <string.h>


int main(int argc, char* argv[])
{
   char *filename2 = argv[1];
   char *maped = "MapedFile";
   int numbers[3];
   double doubles[3];
   double res1, res2;
   int fd;
   char *ptr;

   fd = open(maped,  O_RDONLY  , 00400);
   if(fd == -1){
      printf("Error file descriptor %s\n", strerror(errno));
      exit(1);
   }
   ptr = mmap(NULL, 3*sizeof(int), PROT_READ, MAP_SHARED, fd, 0);
   if(ptr == MAP_FAILED){
      printf("Map failed in read process: %s\n", strerror(errno));
      exit(1);
   }
   int k = 0;
   for(int i = 0; i < sizeof(numbers) ;i++) {
      if (ptr[i] != 0) {
         numbers[k] = ptr[i];
         k++;
      }
   }

   if((numbers[1] == 0)||(numbers[2] == 0)){
      return -1;
   }
   for(int i = 0; i < 3 ;i++) {
      doubles[i] = numbers[i];
   }

   res1 = doubles[0] / doubles[1];
   res2 = res1 / doubles[2];

   FILE *file = fopen(filename2,"w");
   fprintf(file,"%f", res2);
   fclose(file);
   close(fd);
```

```
    return 0;
}
```

4 Демонстрация работы программы

```
magic@magical:~/CLionProjects/os4$ ./main
letsgo
45 5 3
magic@magical:~/CLionProjects/os4$ cat letsgo
3.000000
magic@magical:~/CLionProjects/os4$ ./main
letsgo2
76 8 9
magic@magical:~/CLionProjects/os4$ cat letsgo
1.055556
```

5 Strace

```
magic@magical:~/CLionProjects/os4$ strace -f -e trace="%process,read,write,dup2,mmap"
-o strace_log.txt ./main
letsgo3
150 6 5
magic@magical:~/CLionProjects/os4$ cat letsgo3
-3.533333
magic@magical:~/CLionProjects/os4$ cat strace_log.txt
5940 execve("./main", ["./main"], 0x7ffc3df90288 /* 68 vars */) = 0
5940 arch_prctl(0x3001 /* ARCH_??? */, 0x7fffa7ac9750) = -1 EINVAL (Invalid argument)
5940 mmap(NULL, 102741, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f10d0ea1000
5940 read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360q\2\0\0\0\0\0"...,
832) = 832
5940 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f10d0e9f000
5940 mmap(NULL, 2036952, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f10d0cad000
5940 mmap(0x7f10d0cd2000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x25000) = 0x7f10d0cd2000
5940 mmap(0x7f10d0e4a000, 303104, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x19d000) = 0x7f10d0e4a000
5940 mmap(0x7f10d0e95000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1e7000) = 0x7f10d0e95000
5940 mmap(0x7f10d0e9b000, 13528, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7f10d0e9b000
5940 arch_prctl(ARCH_SET_FS, 0x7f10d0ea0540) = 0
5940 read(0, "letsgo2\n", 1024)         = 8
5940 read(0, "150 6 5\n", 1024)         = 8
5940 mmap(NULL, 12, PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f10d0ee7000
5940 execve("child", [" ", "letsgo2"], 0x7fffa7ac9838 /* 68 vars */) = 0
5940 arch_prctl(0x3001 /* ARCH_??? */, 0x7ffd06c26fd0) = -1 EINVAL (Invalid argument)
5940 mmap(NULL, 102741, PROT_READ, MAP_PRIVATE, 4, 0) = 0x7ffa81e90000
5940 read(4, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360q\2\0\0\0\0\0"...,
832) = 832
5940 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7ffa81e8e000
5940 mmap(NULL, 2036952, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 4, 0) = 0x7ffa81c9c000
5940 mmap(0x7ffa81cc1000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 4, 0x25000) = 0x7ffa81cc1000
5940 mmap(0x7ffa81e39000, 303104, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4,
0x19d000) = 0x7ffa81e39000
5940 mmap(0x7ffa81e84000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 4, 0x1e7000) = 0x7ffa81e84000
5940 mmap(0x7ffa81e8a000, 13528, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7ffa81e8a000
5940 arch_prctl(ARCH_SET_FS, 0x7ffa81e8f540) = 0
5940 mmap(NULL, 12, PROT_READ, MAP_SHARED, 4, 0) = 0x7ffa81ed6000
5940 write(5, "-3.533333", 9)    = 9
5940 exit_group(0)                = ?
5940 +++ exited with 0 +++
```

6 Выводы

В ходе решения данной лабораторной работы я научился работать с file-mapping-ом и отработал его применение на практике. Также я ещё раз попрактиовался работе с файлами. В целом я считаю полезными полученные знания, так как file mapping является полезным для ускорения работы программы механизмом межпроцессорного взаимодействия. В дополнение к ускореию можно добавить то, что можно не запоминать расположение файла и то, что сдвиг при открывании файла не требует допольнительных системных вызовов.