Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №4 по курсу «Операционные системы»
Отображение файлов в память

Студент: А. П. Шорохов
Преподаватель: А. А. Соколов
Группа: М8О-206Б
Дата:
Оценка:
Подпись:

Москва, 2020

Лабораторная работа №4

Задача: 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).

Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

Вариант: Родительский процесс представляет собой сервер по работе с массивами и принимает команды со стороны дочернего процесса.

1 Описание

Функция mkstemp() создает временный файл с уникальным именем. mmap() возвращает адрес начала строки, которая будет использована в обоих процессах.
Дочерний процесс принимает на вход файл с данными, после чего записывает их в память. Родительский процесс в свою очередь умножает все переданные числа на 10 и пишет их в выходной файл.

2 Исходный код

```
1
2
3  #include <stdio.h>
4  #include <sys/mman.h>
5  #include <sys/stat.h>
6  #include <fcntl.h>
7  #include <stdlib.h>
8  #include <sys/types.h>
9  #include <unistd.h>
10 #include <ctype.h>
11 #include <sys/wait.h>
12 #include <semaphore.h>
13
14 const int BUFFER_SIZE = 51;
15
16 int create_tmp() {
17
18   char tmp_name[] = "/tmp/tmpf.XXXXXX";
19   int tmp_fd = mkstemp(tmp_name);
20   if ( tmp_fd == -1) {
21     printf("error\n");
22     exit(1);
23   }
24   int size = BUFFER_SIZE + 1;
25   char array[size];
26   for ( int i = 0; i < size; ++i ) {
27     array[i] = '\0';
28   }
29   write(tmp_fd, array, size);
30   return tmp_fd;
31
32 }
33
34
35 int main(int argc, char * argv[]) {
36
37
38   int fd = create_tmp();
39   int* memory1 = (int*) mmap(NULL,10,PROT_WRITE | PROT_READ, MAP_SHARED, fd, 0);
40   if (memory1 == NULL) {
41     printf("error mapping\n");
42     exit(1);
43   }
44
45   const char* out_sem_name = "/o_s";
46   sem_unlink(out_sem_name);
47   sem_t* out = sem_open(out_sem_name, O_CREAT, 777, 0);
```

```
48
49  pid_t childPID = fork();
50
51  if (childPID == 0) { // client
52    printf("client\n");
53
54    FILE * read = fopen("input.txt", "r");
55
56    int size;
57
58    fscanf(read, "%d", &size);
59
60    int array[size];
61
62    for (int i = 0; i < size; i++) {
63      fscanf(read, "%d", &array[i]);
64    }
65
66    memory1[0] = size;
67    for (int i = 0; i < size; i++) {
68      memory1[i + 1] = array[i];
69    }
70
71    fclose(read);
72
73    sem_post(out);
74    exit(0);
75  } else if (childPID > 0) {
76    sem_wait(out);
77    printf("server\n");
78
79    int size = memory1[0];
80
81
82    int array[size];
83
84    for (int i = 1; i <= size; i++) {
85      array[i - 1] = memory1[i];
86      array[i - 1] *= 10;
87    }
88
89    FILE * write = fopen("output.txt", "w");
90
91    for (int i = 0; i < size; i++) {
92      fprintf(write, "%d ", array[i]);
93    }
94
95    fclose(write);
96
```

```
97    }
98    else {
99      fprintf(stderr, "fork() was not finished successfully");
100     return 1;
101   }
102
103
104   return 0;
105 }
```

3 Тестирование

Входной файл

```
50
7689 7247 3784 3567 1605 -5031 4672 8456 -4004 -7955 6656 -971 4993 9026 -4532
-4047 4835 3352 -7635 4120 5052 6991 7984 3236 7717 -5723 6702 -5854 -9186
-8521 -5607 -9019 -6577 -1120 9248 -3101 -5066 -1324 9875 4283 -3443 9185 2312
-2392 9844 8504 -1219 4988 9730 2850
```

Выходной файл

```
76890 72470 37840 35670 16050 -50310 46720 84560 -40040 -79550 66560 -9710
49930 90260 -45320 -40470 48350 33520 -76350 41200 50520 69910 79840 32360
77170 -57230 67020 -58540 -91860 -85210 -56070 -90190 -65770 -11200 92480 -31010
-50660 -13240 98750 42830 -34430 91850 23120 -23920 98440 85040 -12190 49880
97300 28500
```

4 Выводы

Отображение файлов даёт удобство при работе с файлами, так как позволяет работать с областью файла как с обычным куском памяти. Другими словами, мы имеем доступ к каждому байту области памяти, которую мы отобразили, количество системных вызовов по чтению и записи сводятся к нулю, так как мы работаем с оперативной памятью.