Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №4 по курсу «Операционные системы»

File mapping

Студент: О. В. Бабин
Преподаватель: А. А. Соколов
Группа: М8О-206Б-19
Дата: 25.12.2020
Оценка:
Подпись:

Москва, 2020

1 Постановка задачи

Цель работы:

Приобретение практических навыков в:

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

Задание (вариант 18):

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.

Родительский процесс создает два дочерних процесса. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись для child1. Аналогично для второй строки и процесса child2. Родительский и дочерний процесс должны быть представлены разными программами.

Правило фильтрации: нечетные строки отправляются child1, четные child2. Дочерние процессы удаляют все гласные из строк.

2 Общие сведения о программе

Программа компилируется из файла main.c. Отличительной особенностью программы (относительно кода лабораторной №2) является использование системного вызова mmap. Этот системный вызов запускается с ключом MAP_ANONYMOUS, это означает, что файл, отображаемый в память, будет виртуальным. Также в этой лабораторной используется обработчик системного сигнала (сигнала о завершении работы родительского процесса).

3 Общий метод и алгоритм решения

Для реализации поставленной задачи необходимо:

1. Изучить принципы работы mmap.
2. Изучить принципы работы системных сигналов.
3. Переписать функции, заменив pipe-ы на mmap.
4. Написать обработчики сигналов.
5. Написать обработку ошибок
6. Написать тесты

4 Исходный код

main.c

```
1
2  #include <fcntl.h>
3  #include <signal.h>
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <sys/mman.h>
7  #include <sys/wait.h>
8  #include <unistd.h>
9
10 int is_it_time_to_terminate = 0;
11 const int OFFSET = sizeof(int);
12
13 void update_is_it_time_to_terminate(int num) {
14   is_it_time_to_terminate = 1;
15 }
16
17 void child_work(char* from, int to) {
18   int idx = OFFSET;
19   while (1) {
20     if (idx < OFFSET + ((int*)from)[0]) {
21       char c = from[idx];
22       if (c != 'a' && c != 'e' && c != 'i' && c != 'o' && c != 'u' && c != 'y' &&
23           c != 'A' && c != 'E' && c != 'I' && c != 'O' && c != 'U' && c != 'Y') {
24         write(to, &c, 1);
25       }
26       ++idx;
27     } else {
28       if (is_it_time_to_terminate) {
29         break;
30       }
31     }
32   }
33   close(to);
34 }
35
36 void parrent_work(pid_t child1, char* child_map1, pid_t child2, char* child_map2) {
37   char ch;
38   int is_even = 0;
39   int idx1 = OFFSET;
40   int idx2 = OFFSET;
41   while (read(STDIN_FILENO, &ch, 1) > 0) {
42     if (!is_even) {
43       child_map1[idx1++] = ch;
44     } else {
45       child_map2[idx2++] = ch;
46     }
```

```
47      if (ch == '\n') {
48        if (!is_even) {
49          ((int*)child_map1)[0] = idx1 - OFFSET;
50        } else {
51          ((int*)child_map2)[0] = idx2 - OFFSET;
52        }
53        is_even = !is_even;
54      }
55    }
56
57    kill(child1, SIGUSR1);
58    kill(child2, SIGUSR1);
59    int res1;
60    int res2;
61    waitpid(child1, &res1, 0);
62    waitpid(child2, &res2, 0);
63    if (res1 != 0 || res2 != 0) {
64      fprintf(stderr, "Something ended ne tak!\n%d %d\n", res1, res2);
65    }
66  }
67
68  int read_name_and_open_file() {
69    const size_t FILE_NAME_SIZE = 64;
70    char f_name[FILE_NAME_SIZE];
71    char buf[1];
72    int idx = 0;
73    while (idx < FILE_NAME_SIZE && read(STDIN_FILENO, buf, 1) > 0) {
74      if (buf[0] == '\n') {
75        break;
76      }
77      f_name[idx++] = buf[0];
78    }
79    f_name[idx++] = '\0';
80    return open(f_name, O_WRONLY | O_TRUNC);
81  }
82
83  void error(char* buf, size_t size) { write(STDERR_FILENO, buf, size); }
84  void check_file_id(int id) {
85    if (id == -1) {
86      error("File not found\n", 15);
87      exit(-1);
88    }
89  }
90  void* check_map_creation() {
91    void* m_file = mmap(NULL, 2048, PROT_READ | PROT_WRITE, MAP_SHARED | MAP_ANONYMOUS,
          -1, 0);
92    if (m_file == MAP_FAILED) {
93      error("Cannot create mmap\n", 19);
94      exit(-2);
```

```
95    }
96    ((int*)m_file)[0] = 0;
97    return m_file;
98  }
99  int check_fork() {
100   int fd = fork();
101   if (fd == -1) {
102     error("Cannot create process\n", 22);
103     exit(-3);
104   }
105   return fd;
106 }
107 void add_signals() {
108   void (*func)(int);
109   func = signal(SIGUSR1, update_is_it_time_to_terminate);
110   if (func == SIG_IGN) {
111     error("Cannot add signal\n", 18);
112     exit(-4);
113   }
114 }
115
116 int main(int argc, char* argv[]) {
117   add_signals();
118
119   int f1 = read_name_and_open_file();
120   check_file_id(f1);
121   int f2 = read_name_and_open_file();
122   check_file_id(f2);
123
124   char* m_file1 = check_map_creation();
125   pid_t child1 = check_fork();
126   if (child1 == 0) {
127     close(f2);
128     child_work(m_file1, f1);
129     return 0;
130   }
131   close(f1);
132
133   char* m_file2 = check_map_creation();
134   pid_t child2 = check_fork();
135   if (child2 == 0) {
136     child_work(m_file2, f2);
137     return 0;
138   }
139   close(f2);
140
141   parrent_work(child1, m_file1, child2, m_file2);
142
143   return 0;
```

```
144 || }
```

5 Пример работы

Первый тест - проверка на обработку случая отсутсвия первого файла

```
windicor@Lina-HP:~$ cat test1
f
windicor@Lina-HP:~$ ./a.out <test1
File not found
```

Второй тест - проверка на обработку случая отсутсвия второго файла

```
windicor@Lina-HP:~$ cat test2
f1
f
windicor@Lina-HP:~$ ./a.out <test2
File not found
```

Третий и четвёртый тесты - проверка работоспособности программы на корректных данных

```
windicor@Lina-HP:~$ cat test3
f1
f2
a b c d e f g h i j k l
m n o p q r s t u v w x y z
windicor@Lina-HP:~$ ./a.out <test3
windicor@Lina-HP:~$ cat f1
b c d  f g h  j k l
windicor@Lina-HP:~$ cat f2
m n  p q r s t  v w x  z
windicor@Lina-HP:~$ cat test4
f1
f2
m n o p q r s t u v w x y z
a b c d e f g h i j k l
m n o p q r s t u v w x y z
```

```
a b c d e f g h i j k l
m n o p q r s t u v w x y z
a b c d e f g h i j k l
m n o p q r s t u v w x y z
a b c d e f g h i j k l
windicor@Lina-HP:~$ ./a.out <test4
windicor@Lina-HP:~$ cat f1
m n  p q r s t  v w x  z
m n  p q r s t  v w x  z
m n  p q r s t  v w x  z
m n  p q r s t  v w x  z
windicor@Lina-HP:~$ cat f2
b c d  f g h  j k l
b c d  f g h  j k l
b c d  f g h  j k l
b c d  f g h  j k l
```

Пятый тест - проверка работоспособности программы на пустых данных

```
windicor@Lina-HP:~$ cat test5
f1
f2
windicor@Lina-HP:~$ ./a.out <test5
windicor@Lina-HP:~$ cat f1
windicor@Lina-HP:~$ cat f2
```

6 Вывод

В процессе работы над лабораторной я научился основам работы с файлами отображаемыми в память и системными сигналами в Си. Файлы отображаемые в память являются крайне полезным инструментом, так как в ряде случаем эта технология способна значительно повысить эффективность работы программы. Системные сигналы оказались полезными при работе над последующими лабораторными.