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: В. М. Ватулин
Преподаватель: А. А. Соколов
Группа: М8О-206Б-19
Дата: 17.04.2021
Оценка:
Подпись:

Москва, 2021

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 20):

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.

Правило фильтрации: строки длины больше 10 символов отправляются в pipe2, иначе в pipe1. Дочерние процессы инвертируют строки.

2 Общие сведения о программе

Программа принимает аргументами командной строки имена первого и второго файла, в которые будут записываться результаты. Затем программа принимает в stdin строки, которые направляются замапленную память. После этого родительский процесс посылает сигнал ребенку, для которого предназначена строка (в зависимости от длины строки). Ребенок инвертирует строку, отправляет ее обратно родительскому процессу и сигналом сообщает ему, что строка готова для вывода. Родительский процесс выводит строку. Программа завершается посыланием EOF в stdin.

3 Общий метод и алгоритм решения

Для реализации поставленной задачи необходимо:

1. Изучить основы file mapping'a
2. Изучить принципи работы сигналов в ОС Linux для синхронизации между процессами
3. Создать общую для процессов память
4. Написать функцию родительского процесса
5. Написать функцию дочернего процесса
6. Написать обработку ошибок

4 Исходный код

main.c

```
1
2  #include <stdlib.h>
3  #include <stdio.h>
4  #include <string.h>
5
6  #include <sys/mman.h>
7  #include <unistd.h>
8  #include <signal.h>
9  #include <sys/prctl.h>
10
11 #define BUFFSIZE 256
12 #define PROC_NUM 2
13
14 sig_atomic_t sigs_got = 0;
15
16 void wait_sig(int sig) {
17     ++sigs_got;
18 }
19
20 void parent_death(int sig) {
21     exit(1);
22 }
23
24 void parent_job(char shared_mem[BUFFSIZE + 1], pid_t child_pid[PROC_NUM]) {
25     sigset_t empty_sigset;
26     sigemptyset(&empty_sigset);
27     while (sigs_got != 2) {
28     }
29     sigset_t blocking_sigset;
30     sigemptyset(&blocking_sigset);
31     sigaddset(&blocking_sigset, SIGUSR1);
32     sigprocmask(SIG_BLOCK, &blocking_sigset, NULL);
33     while (fgets(shared_mem, BUFFSIZE + 1, stdin) != NULL) {
34         if (strlen(shared_mem) > 10) {
35             kill(child_pid[1], SIGUSR1);
36         }
37         else {
38             kill(child_pid[0], SIGUSR1);
39         }
40         sigsuspend(&empty_sigset);
41     }
42 }
43
44 void child_job(char *shared_mem, pid_t parent_pid, char *filename) {
45     prctl(PR_SET_PDEATHSIG, SIGTERM);
46     signal(SIGTERM, parent_death);
```

```
47      if (getppid() != parent_pid) {
48          parent_death(SIGTERM);
49      }
50      sigset_t empty_sigset;
51      sigemptyset(&empty_sigset);
52      sigset_t blocking_sigset;
53      sigemptyset(&blocking_sigset);
54      sigaddset(&blocking_sigset, SIGUSR1);
55      sigprocmask(SIG_BLOCK, &blocking_sigset, NULL);
56      FILE *fp = fopen(filename, "w");
57      if (fp == NULL) {
58          perror("Error");
59          kill(parent_pid, SIGINT);
60      }
61      kill(parent_pid, SIGUSR1);
62      while (1) {
63          sigsuspend(&empty_sigset);
64          for (int i = strlen(shared_mem) - 2; i >= 0; --i) {
65              fputc(shared_mem[i], fp);
66          }
67          fputc('\n', fp);
68          kill(parent_pid, SIGUSR1);
69      }
70  }
71
72  int main(int argc, char **argv) {
73      if (argc != 3) {
74          fprintf(stderr, "Error: wrong number of arguments, should be <prog_name> <
                filename1> <filename2>\n");
75          exit(1);
76      }
77      char *shared_mem = mmap(NULL, BUFFSIZE + 1, PROT_READ|PROT_WRITE, MAP_SHARED|
            MAP_ANON, -1, 0);
78      if (shared_mem == MAP_FAILED) {
79          perror("Error");
80          exit(1);
81      }
82      signal(SIGUSR1, wait_sig);
83      char *filename;
84      pid_t child_pid[PROC_NUM];
85      pid_t parent_pid = getpid();
86      pid_t temp_pid = 1;
87      for (int i = 0; i < PROC_NUM; ++i) {
88          if (temp_pid != 0) {
89              temp_pid = fork();
90              filename = argv[i + 1];
91              child_pid[i] = temp_pid;
92              if (temp_pid == -1) {
93                  perror("fork error");
```

```
94                exit(1);
95            }
96        }
97        else {
98            break;
99        }
100    }
101    if (temp_pid == 0) {
102        child_job(shared_mem, parent_pid, filename);
103    }
104    else {
105        parent_job(shared_mem, child_pid);
106    }
107
108    return 0;
109}
```

5 Пример работы

```
eri412@Eri-PC:~/Desktop/study/OS/OSlab4$ ./a.out 1.txt 2.txt
qwe
rty
qweqweqeqeqwqe
12311
asdadasdasdasd5345
65363636345353
sadadxzc
eri412@Eri-PC:~/Desktop/study/OS/OSlab4$ cat 1.txt
ewq
ytr
11321
czxdadas
eri412@Eri-PC:~/Desktop/study/OS/OSlab4$ cat 2.txt
eqwqeqeqewqewq
5435dsadsadsadadsa
35354363636356
```

6 Вывод

В этой лабораторной я узнал еще один способ передачи данных при взаимодействии процессов. Я понял как можно создавать отображения физических файлов, а также делать анонимные отображения. Так же я более полно ознакомился с принципом работы системы сигналов в ОС Linux.