Лабораторная работа № 4 по курсу : Операционные системы

Выполнил студент группы М8О-206Б-17 МАИ *Новиков Павел Сергеевич.*

Цель работы

Приобретение практических навыков в:

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

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.

Задание

Вариант

Поиск длин кратчайших путей в ориентированом взвешеном графе

Информация

Отличие от второй лабораторной лишь в том, что нужно использовать mmap для отображения виртуальной памяти.

Исходный код

```
#include <stdio.h>
#include <ctype.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <string.h>
#include <sys/mman.h>
#include <semaphore.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <signal.h>
#include <stdio.h>
```

```c
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>

int *mapped_memory;
#define READ    0
#define WRITE   1

#define SIZE sizeof(int)


int factorial (int n) {
    if (n == 0) return 1;
    pid_t pid;
    int res;
    pid = fork();
    if (pid < 0) {
        perror("fork");
    } else if (pid > 0) {
        wait(0);
        res = *mapped_memory;
    } else if (pid == 0) {
        res = factorial(n - 1);
        *mapped_memory = res;
        exit(0);
    }
    return n * res;
}

int main() {

    int fd = shm_open("/tmp:memory", O_RDWR | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);
        if(fd == -1){
                perror("shm::open_fail");
                exit(-1);
        }
        if(ftruncate(fd, SIZE) == -1){
                perror("trucate::fail");
                exit(-1);
        }

        mapped_memory = mmap(NULL, SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
        if(mapped_memory == MAP_FAILED){
                perror("mmap::mapping_fail");
                fprintf(stderr, "%p", mapped_memory);
                exit(-1);
        }
        close(fd);

    int n = 0;

    printf("Enter␣N␣>␣0␣to␣calculate␣Factorial(N)\n");
    scanf("%d", &n);
    if (n < 0) {
        printf("Bad␣input\n");
        return 0;
    }
    printf("Factorial(N)␣equals:\n%d\n", factorial(n));

    return 0;
}
```

Системные вызовы:

1. **int** shm_open(**const char** *name, **int** oflag, mode_t mode); – создаёт новый разделяемый файл, который может предоставить доступ на чтение и запись сразу нескольким процессам. При успехе возвращает файловый дескриптор на соответствующий файл, при неудаче -1.

   Были использованы следующие флаги:

   O_RDWR Open the object for read-write access.

   O_CREAT Create the shared memory object if it does not exist.

   O_TRUNC If the shared memory object already exists, truncate it to zero bytes.

2. **int** ftruncate(**int** fildes , off_t length); – устанавливает размер файла в length байт. При успехе возвращает 0. Если указан неверный дескриптор, или дескриптор указывает на директорию, или возникла другая ошибка возвращается -1.

3. **void** *mmap(**void** *addr, size_t len, **int** prot, **int** flags, **int** fildes , off_t off);

   – устанавливает отображение между адресным пространством процесса и объектом памяти.

   Были использованы следующие флаги:

   PROT_READ Data can be read.

   PROT_WRITE Data can be written.

   MAP_SHARED Changes are shared.

4. **int** munmap(**void** *addr, size_t len); – удаляет все сопоставления для всех страниц, содержащих любую часть адресного пространства процесса, начиная с addr и на len байт.

Тесты

```
tilt@tilt:~/os/lab04$ ./prog
Enter N > 0 to calculate Factorial(N)
5
Factorial(N) equals:
120
tilt@tilt:~/os/lab04$ ./prog
Enter N > 0 to calculate Factorial(N)
7
Factorial(N) equals:
5040
```

Вывод strace

```
mmap(NULL, 218437, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f2c9252d000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x
mmap(NULL, 39416, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f2c9252100
mmap(0x7f2c92523000, 16384, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DE
mmap(0x7f2c92527000, 8192, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
mmap(0x7f2c92529000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DE
mmap(NULL, 1852992, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f2c9235c
mmap(0x7f2c9237e000, 1359872, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_
mmap(0x7f2c924ca000, 311296, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
mmap(0x7f2c92517000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_D
mmap(0x7f2c9251d000, 13888, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_A
mmap(NULL, 131528, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f2c9233b0
mmap(0x7f2c92341000, 61440, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DE
mmap(0x7f2c92350000, 24576, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3
mmap(0x7f2c92356000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DE
mmap(0x7f2c92358000, 12744, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_A
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
mmap(NULL, 4, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f2c9258c000
Enter N > 0 to calculate Factorial(N)
4
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=19219, si_uid=1000,
Factorial(N) equals:
24
+++ exited with 0 +++
Time: 0h:00m:08s
```

Выводы

Были приобретены минимальные навыки для работы с разделяемой памятью, наверно более практичный способ межпроцессного общения.