Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

Лабораторная работа №3 по курсу

«Операционные системы»

Группа: М8О-210Б-23

Студент: Коваленко Д.А

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 26.12.24

Москва, 2024

Постановка задачи

Вариант 13.

Родительский процесс создает дочерний процесс. Первой строкой пользователь вводит имя файла через консоль родительского процесса. Файл открывается для чтения и перенаправляется в стандартный поток ввода дочернего процесса. Дочерний процесс считывает строки из этого файла, конвертирует их в верхний регистр и выводит в pipe1. Родительский процесс получает данные из pipe1, выполняет подсчет количества слов в каждой строке и выводит результат в консоль. Родительский и дочерний процессы должны быть представлены разными программами.

В файле находятся строки вида: «текст текст текст». Дочерний процесс должен преобразовывать символы каждой строки в верхний регистр. Родительский процесс должен считать количество слов в каждой полученной строке и выводить его вместе с преобразованной строкой. Числа, записанные в файле, игнорируются при подсчете слов.

Если входной файл пуст, дочерний и родительский процессы завершают свою работу без дополнительных действий.

Общий метод и алгоритм решения

Программа организует параллельную обработку данных через общую память (shared memory) и семафоры.

1. **Основной процесс:**
    - Создает общую память и семафоры.
    - Запускает два дочерних процесса.
    - Читает строки из файла, отправляет их в общую память и синхронизирует обработку через семафоры.
    - Сохраняет обработанные строки в выходной файл.
2. **Первый дочерний процесс:**
    - Читает строку из памяти, преобразует символы в нижний регистр и возвращает результат.
3. **Второй дочерний процесс:**

- Читает строку из памяти, заменяет пробелы на подчеркивания и возвращает результат.

4. **Системные вызовы:**
   - `shm_open`, `mmap`, `munmap` — работа с общей памятью.
   - `sem_open`, `sem_wait`, `sem_post` — синхронизация через семафоры.
   - `fork` — создание процессов.
   - `waitpid` — ожидание завершения процессов.

В результате данные обрабатываются параллельно, улучшая производительность.

Код программы

main/main.cpp

```
#include "error_handling.h"
#include <cerrno>
#include <cstring>
#include <fcntl.h>
#include <semaphore.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>


const int SHM_SIZE = 4096;
const char *SHM_NAME = "/shm_example";
const char *SEM_NAME_READ = "/sem_read";
const char *SEM_NAME_WRITE = "/sem_write";


int main() {
  char file_path[1024];


  write(STDOUT_FILENO, "Введите путь к файлу: ", 24);
  ssize_t bytes_read = read(STDIN_FILENO, file_path, sizeof(file_path) - 1);


  if (bytes_read < 0) {
    myPerror("Ошибка при чтении пути к файлу");
    _exit(errno);
  }
```

```
if (bytes_read > 0 && file_path[bytes_read - 1] == '\n') {
  file_path[bytes_read - 1] = '\0';
}

int shm_fd = shm_open(SHM_NAME, O_CREAT | O_RDWR, 0666);
if (shm_fd == -1) {
  myPerror("Ошибка при создании shared memory");
  _exit(errno);
}

if (ftruncate(shm_fd, SHM_SIZE) == -1) {
  myPerror("Ошибка при установке размера shared memory");
  _exit(errno);
}

void *shm_ptr =
    mmap(0, SHM_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, shm_fd, 0);
if (shm_ptr == MAP_FAILED) {
  myPerror("Ошибка при отображении shared memory");
  _exit(errno);
}

sem_t *sem_read = sem_open(SEM_NAME_READ, O_CREAT, 0666, 0);
sem_t *sem_write = sem_open(SEM_NAME_WRITE, O_CREAT, 0666, 1);

if (sem_read == SEM_FAILED || sem_write == SEM_FAILED) {
  myPerror("Ошибка при создании семафоров");
  _exit(errno);
}

pid_t pid1 = fork();
if (pid1 == 0) {
  execlp("./child1_process", "./child1_process", nullptr);
  myPerror("Ошибка при выполнении exec для child1");
  _exit(errno);
}
```

```
pid_t pid2 = fork();
if (pid2 == 0) {
  execlp("./child2_process", "./child2_process", nullptr);
  myPerror("Ошибка при выполнении exec для child2");
  _exit(errno);
}

int file_fd = open(file_path, O_RDONLY);
if (file_fd == -1) {
  myPerror("Ошибка при открытии файла");
  return 1;
}

char input_buffer[1024];
ssize_t file_bytes_read;

while ((file_bytes_read = read(file_fd, input_buffer, sizeof(input_buffer))) >
      0) {
  sem_wait(sem_write);

  memcpy(shm_ptr, input_buffer, file_bytes_read);
  ((char *)shm_ptr)[file_bytes_read] = '\0';
  sem_post(sem_read);

  sem_wait(sem_write);
  write(STDOUT_FILENO, shm_ptr, strlen((char *)shm_ptr));
  sem_post(sem_read);
}

if (file_bytes_read < 0) {
  myPerror("Ошибка при чтении файла");
}

close(file_fd);

waitpid(pid1, nullptr, 0);
```

```
  waitpid(pid2, nullptr, 0);

  munmap(shm_ptr, SHM_SIZE);
  shm_unlink(SHM_NAME);

  sem_close(sem_read);
  sem_close(sem_write);
  sem_unlink(SEM_NAME_READ);
  sem_unlink(SEM_NAME_WRITE);

  return 0;
}
```

Child1_process.cpp

```
#include "error_handling.h"
#include <cctype>
#include <cerrno>
#include <cstring>
#include <fcntl.h>
#include <semaphore.h>
#include <sys/mman.h>
#include <unistd.h>

const int SHM_SIZE = 4096;
const char *SHM_NAME = "/shm_example";
const char *SEM_NAME_READ = "/sem_read";
const char *SEM_NAME_WRITE = "/sem_write";

int main() {
  int shm_fd = shm_open(SHM_NAME, O_RDWR, 0666);
  if (shm_fd == -1) {
    myPerror("Ошибка при открытии shared memory в child1");
    _exit(errno);
  }

  void *shm_ptr =
      mmap(0, SHM_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, shm_fd, 0);
```

```
 if (shm_ptr == MAP_FAILED) {
  myPerror("Ошибка при отображении shared memory в child1");
  _exit(errno);
 }

 sem_t *sem_read = sem_open(SEM_NAME_READ, 0);
 sem_t *sem_write = sem_open(SEM_NAME_WRITE, 0);

 while (true) {
  sem_wait(sem_read);
  char *data = (char *)shm_ptr;

  for (int i = 0; data[i] != '\0'; ++i) {
   data[i] = tolower(data[i]);
  }

  sem_post(sem_write);
 }

 return 0;
}
```

Child2_process.cpp

```
#include "error_handling.h"
#include <cctype>
#include <cerrno>
#include <cstring>
#include <fcntl.h>
#include <semaphore.h>
#include <sys/mman.h>
#include <unistd.h>

const int SHM_SIZE = 4096;
const char *SHM_NAME = "/shm_example";
const char *SEM_NAME_READ = "/sem_read";
const char *SEM_NAME_WRITE = "/sem_write";

int main() {
```

```
int shm_fd = shm_open(SHM_NAME, O_RDWR, 0666);
if (shm_fd == -1) {
  myPerror("Ошибка при открытии shared memory в child2");
  _exit(errno);
}

void *shm_ptr =
    mmap(0, SHM_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, shm_fd, 0);
if (shm_ptr == MAP_FAILED) {
  myPerror("Ошибка при отображении shared memory в child2");
  _exit(errno);
}

sem_t *sem_read = sem_open(SEM_NAME_READ, 0);
sem_t *sem_write = sem_open(SEM_NAME_WRITE, 0);

while (true) {
  sem_wait(sem_read);

  char *data = (char *)shm_ptr;

  for (int i = 0; data[i] != '\0'; ++i) {
    if (isspace(data[i]) && data[i] != '\n') {
      data[i] = '_';
    }
  }

  sem_post(sem_write);
}

return 0;
}
```

Протокол работы программы

strace ./main

execve("./main", ["./main"], 0x7ffde4a285a0 /* 89 vars */) = 0

brk(NULL)   = 0x61ad939cf000

arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe46fe7e10) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7e1d5a83b000

access("/etc/ld.so.preload", R_OK)   = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/snap/alacritty/140/usr/lib/x86_64-linux-gnu/dri/glibc-hwcaps/x86-64-v3/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/snap/alacritty/140/usr/lib/x86_64-linux-gnu/dri/glibc-hwcaps/x86-64-v3", 0x7ffe46fe7030, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/snap/alacritty/140/usr/lib/x86_64-linux-gnu/dri/glibc-hwcaps/x86-64-v2/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/snap/alacritty/140/usr/lib/x86_64-linux-gnu/dri/glibc-hwcaps/x86-64-v2", 0x7ffe46fe7030, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/snap/alacritty/140/usr/lib/x86_64-linux-gnu/dri/tls/x86_64/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/snap/alacritty/140/usr/lib/x86_64-linux-gnu/dri/tls/x86_64/x86_64", 0x7ffe46fe7030, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/snap/alacritty/140/usr/lib/x86_64-linux-gnu/dri/tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/snap/alacritty/140/usr/lib/x86_64-linux-gnu/dri/tls/x86_64", 0x7ffe46fe7030, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/snap/alacritty/140/usr/lib/x86_64-linux-gnu/dri/tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/snap/alacritty/140/usr/lib/x86_64-linux-gnu/dri/tls/x86_64", 0x7ffe46fe7030, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/snap/alacritty/140/usr/lib/x86_64-linux-gnu/dri/tls/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/snap/alacritty/140/usr/lib/x86_64-linux-gnu/dri/tls", 0x7ffe46fe7030, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/snap/alacritty/140/usr/lib/x86_64-linux-gnu/dri/x86_64/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/snap/alacritty/140/usr/lib/x86_64-linux-gnu/dri/x86_64/x86_64", 0x7ffe46fe7030, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/snap/alacritty/140/usr/lib/x86_64-linux-gnu/dri/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/snap/alacritty/140/usr/lib/x86_64-linux-gnu/dri/x86_64", 0x7ffe46fe7030, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/snap/alacritty/140/usr/lib/x86_64-linux-gnu/dri/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/snap/alacritty/140/usr/lib/x86_64-linux-gnu/dri/x86_64", 0x7ffe46fe7030, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/snap/alacritty/140/usr/lib/x86_64-linux-gnu/dri/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/snap/alacritty/140/usr/lib/x86_64-linux-gnu/dri", {st_mode=S_IFDIR|0755, st_size=288, ...}, 0) = 0

openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=105007, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 105007, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7e1d5a821000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0\0"..., 832) = 832

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0I\17\357\204\3$\f\221\2039x\
324\224\323\236S"..., 68, 896) = 68

newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...},
AT_EMPTY_PATH) = 0

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"...,
784, 64) = 784

**mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x7e1d5a400000**

mprotect(0x7e1d5a428000, 2023424, PROT_NONE) = 0

**mmap(0x7e1d5a428000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) =
0x7e1d5a428000**

**mmap(0x7e1d5a5bd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1bd000) = 0x7e1d5a5bd000**

**mmap(0x7e1d5a616000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x7e1d5a616000**

**mmap(0x7e1d5a61c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7e1d5a61c000**

close(3)   = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_ANONYMOUS, -1, 0) = 0x7e1d5a81e000

arch_prctl(ARCH_SET_FS, 0x7e1d5a81e740) = 0

set_tid_address(0x7e1d5a81ea10)   = 90021

set_robust_list(0x7e1d5a81ea20, 24)   = 0

rseq(0x7e1d5a81f0e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7e1d5a616000, 16384, PROT_READ) = 0

mprotect(0x61ad91bc0000, 4096, PROT_READ) = 0

mprotect(0x7e1d5a875000, 8192, PROT_READ) = 0

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0

munmap(0x7e1d5a821000, 105007)   = 0

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \
320\277\321\203\321\202\321\214 ", 24Введите путь) = 24

read(0, ./tes1.txt

"./tes1.txt\n", 1023) = 11

openat(AT_FDCWD, "/dev/shm/shm_example", O_RDWR|O_CREAT|
O_NOFOLLOW|O_CLOEXEC, 0666) = 3

ftruncate(3, 4096) = 0

**mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) =
0x7e1d5a874000**

openat(AT_FDCWD, "/dev/shm/sem.sem_read", O_RDWR|O_NOFOLLOW) = 4

newfstatat(4, "", {st_mode=S_IFREG|0664, st_size=32, ...}, AT_EMPTY_PATH)
= 0

getrandom("\x59\xab\xa1\x51\x3d\x20\xfd\x57", 8, GRND_NONBLOCK) = 8

brk(NULL) = 0x61ad939cf000

brk(0x61ad939f0000) = 0x61ad939f0000

**mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) =
0x7e1d5a83a000**

close(4) = 0

openat(AT_FDCWD, "/dev/shm/sem.sem_write", O_RDWR|O_NOFOLLOW) = 4

newfstatat(4, "", {st_mode=S_IFREG|0664, st_size=32, ...}, AT_EMPTY_PATH)
= 0

mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) =
0x7e1d5a839000

close(4) = 0

clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|
CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x7e1d5a81ea10) = 90026

clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|
CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x7e1d5a81ea10) = 90027

openat(AT_FDCWD, "./tes1.txt", O_RDONLY) = 4

read(4, "123QWEWQe ewqwqWEQWE\nrqwrqwWQRWQ"..., 1024) = 48

futex(0x7e1d5a83a000, FUTEX_WAKE, 1) = 0

futex(0x7e1d5a839000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
NULL, FUTEX_BITSET_MATCH_ANY) = 0

write(1, "123QWEWQe_ewqwqWEQWE\nrqwrqwWQRWQ"..., 48123QWEWQe_ewqwqWEQWE

rqwrqwWQRWQr_____rqwrqr

) = 48

futex(0x7e1d5a83a000, FUTEX_WAKE, 1) = 1

read(4, "", 1024) = 0

close(4) = 0

wait4(90026, ^CNULL, 0, NULL) = ? ERESTARTSYS (To be restarted if SA_RESTART is set)

strace: Process 90021 detached

Вывод:

Лабораторная работа 1 успешно переписана с использованием shared memory