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: Попов А.В.

Преподаватель: Бахарев В.Д.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 24.12.24

Москва, 2024

**Постановка задачи**

**Вариант 21.**

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.

Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1 или в pipe2 в зависимости от правила фильтрации. Процесс child1 и child2 производят работу над строками. Процессы пишут результаты своей работы в стандартный вывод.

Правило фильтрации: нечетные строки отправляются в pipe1, четные в pipe2. Дочерние процессы инвертируют строки.

Условие такое: нужно взять свою первую лабу и переделать её с использованием shared memory и memory mapping. Варианты остаются те же, что и у первой лабораторной.

Так как блокирующего чтения из каналов у вас больше не будет, то для синхронизации чтения и записи из shared memory будем использовать семафор.

Для тех, кто будет писать под Windows, прилагаю список системных функций, которые вам понадобятся: CreateFileMapping, MapViewOfFile, UnmapViewOfFile, OpenSemaphore, WaitForSingleObject, ReleaseSemaphore.

Аналогично, для Linux: shm\_open, shm\_unlink, ftruncate, mmap, munmap, sem\_open, sem\_wait, sem\_post, sem\_unlink, sem\_close.

Замерять производительность по сравнению с первой лабораторной работой не нужно.

В отчёте xtrace выделить использование системных вызовов выше.

**Общий метод и алгоритм решения**

Использованные системные вызовы:

* pid\_t fork(void); – создает дочерний процесс.
* ssize\_t write(int fd, const void buf[.count], size\_t count); - пишет count байтов из буфера в файл, на который ссылается файловый дескриптор fd
* ssize\_t read(int fd, void buf[.count], size\_t count) – пытается прочитать count байтов из файлового дескриптора fd в буфер buff
* pid\_t getpid(void); - получить pid текущего или родительского процесса
* int open(const char \*pathname, int flags, mode\_t mode ); - открыть файл с указанными флагами или создать, если указаны специальные флаги, возвращает файловый дескриптор
* int close(int fd); - закрывает файловый дескриптор
* int execv(const char \*pathname, char \*const argv[]); - заменяет образ текущего процесса на новый образ, создается новый стек, кучу и сегменты данных
* pid\_t waidpid (pid\_t pid, int \*stat\_loc, int options); - ждет пока процесс с pid завершится и получается код выхода
* getpid(): Возвращает идентификатор текущего процесса.
* shm\_open(): Открывает или создает объект разделяемой памяти.
* ftruncate(): Изменяет размер файла.
* mmap(): Отображает файл или устройство в память.
* munmap(): Удаляет отображение памяти.
* sem\_open(): Открывает или создает именованный семафор.
* sem\_close(): Закрывает именованный семафор.
* sem\_wait(): Уменьшает значение семафора, блокируя, если значение равно нулю.
* sem\_post(): Увеличивает значение семафора.
* unlink(): Удаляет имя файла.
* waitpid(): Ожидает завершения дочернего процесса.

Программа получает из стандартного ввода имена файлов, создает для каждого из детей pipe и создает дочерний процесс с помощью fork. Перед запуском кода дочернего процесса создаётся буфер для каждого из дочерних процессов и семафор для синхронизации. Далее в дочерний процесс через аргументы передается имя файла для записи и код дочернего запускается через execv. Данные операции повторяются еще раз для второго дочернего процесса. Родительский процесс получается строку из стандартного ввода и пересылает ее в буфер в зависимости от четности строки, увеличивает семафор. Когда на вход поступает EOF или пустая строка, дочерние процессы завершаются, а родительский процесс ждет их заверения. Все процессы закрывают в конце закрывают открытые файлы, семафоры, удаляет общую память

**Код программы**

**parent.c**

#include <stdlib.h>  
#include <unistd.h>  
  
  
#include <sys/wait.h>  
#include <libio/io.h>  
#include <parent/processes.h>  
  
#define MAX\_LINE\_LENGTH 1024  
  
  
  
  
int main(void) {  
 // Get child exec path from environment variable  
 const char \*env\_var\_name = "CHILD\_EXEC\_PATH";  
 char \*exec\_path = getenv(env\_var\_name);  
 if (exec\_path == NULL) {  
 print\_fd(STDERR\_FILENO, "%s: environment variable %s not set\n", exec\_path, env\_var\_name);  
 exit(EXIT\_FAILURE);  
 }  
  
 child\_t child[2] = {  
 create\_empty\_child("child1"),  
 create\_empty\_child("child2"),  
 };  
 const size\_t child\_len = sizeof(child) / sizeof(child[0]);  
  
 // Read file path from stdin  
 for (int i = 0; i < child\_len; i++) {  
 ssize\_t read\_bytes = reads\_fd(STDIN\_FILENO, child[i].file\_path, PATH\_MAX);  
 if (read\_bytes == -1) {  
 print\_fd(STDERR\_FILENO, "Error: Failure during reading file path");  
 exit(EXIT\_FAILURE);  
 }  
 child[i].file\_path[read\_bytes - 1] = '\0'; // Remove trailing newline  
 }  
  
 // Start child processes  
 for (int i = 0; i < child\_len; i++) {  
 const pid\_t status = start\_child\_process(exec\_path, &child[i]);  
 if (status == -1) {  
 exit(EXIT\_FAILURE);  
 }  
 if (status == 0) {  
 // Child process  
 exit(EXIT\_SUCCESS);  
 }  
 }  
  
 // Send to child processes  
 char line[MAX\_LINE\_LENGTH];  
 int line\_number = 1;  
 while (reads\_fd(STDIN\_FILENO, line, MAX\_LINE\_LENGTH) > 0) {  
 if (line[0] == '\n') {  
 // Close on empty line  
 for (int i = 0; i < child\_len; i++) {  
 if (shared\_put\_str(child[i].shared, line) == -1) {  
 print\_fd(STDERR\_FILENO, "Error: Failure during writing to pipe for %s\n", child[i].name);  
 exit(EXIT\_FAILURE);  
 }  
 }  
 break;  
 }  
 child\_t current\_child = child[line\_number % child\_len];  
 if (shared\_put\_str(current\_child.shared, line) == -1) {  
 print\_fd(STDERR\_FILENO, "Error: Failure during writing to pipe for %s\n", current\_child.name);  
 exit(EXIT\_FAILURE);  
 }  
 line\_number++;  
 }  
 // Close all pipes and wait for children  
 for (int i = 0; i < child\_len; i++) {  
 close\_child\_process(child[i]);  
 }  
 return 0;  
}

**processes.c**

*/\*\*  
 \* @file  
 \* @brief  
 \* @details  
 \* @author xsestech   
 \* @date 28.10.2024  
 \*/*#include <parent/processes.h>  
  
child\_t create\_empty\_child(const char\* name) {  
 child\_t child;  
 strncpy(child.name, name, sizeof(child.name));  
 child.pid = -1;  
 return child;  
}  
  
pid\_t start\_child\_process(char \*exec\_path, child\_t \*child) {  
 shared\_handle\_t shared = shared\_producer\_open(child->name, PROC\_MEM\_SIZE);  
 if (shared == NULL) {  
 print\_fd(STDERR\_FILENO, "Error during pipe creation for %s", child->name);  
 return -1;  
 }  
 child->shared = shared;  
  
 pid\_t fork\_pid = fork();  
  
 if (fork\_pid == -1) {  
 print\_fd(STDERR\_FILENO, "Error during creating process for %s\n", child->name);  
 return -1;  
 }  
  
 if (fork\_pid == 0) {  
 // We are child  
 pid\_t child\_pid = getpid();  
  
 printf("%s: child pid %d\n", exec\_path, child\_pid);  
 const char \*args[] = {  
 child->name,  
 child->file\_path,  
 NULL,  
 };  
 const int32\_t status = execv(exec\_path, args);  
  
 if (status != 0) {  
 print\_fd(STDERR\_FILENO, "Error executing %s in %s, status %d \n", exec\_path, child->name, status);  
 return -1;  
 }  
 return 0;  
 }  
 const pid\_t parent\_pid = getppid();  
 child->pid = fork\_pid;  
 print\_fd(STDOUT\_FILENO, "Parent %d: created child with pid %d\n", parent\_pid, fork\_pid);  
 return fork\_pid;  
}  
  
void close\_child\_process(const child\_t child) {  
 shared\_close(child.shared);  
 int child\_status;  
 waitpid(child.pid, &child\_status, 0);  
 print\_fd(STDOUT\_FILENO, "Child %d: exit status %d\n", child.pid, child\_status);  
}

**io.c**

/\*\*  
 \* @file  
 \* @brief  
 \* @details  
 \* @author xsestech   
 \* @date 27.10.2024  
 \*/  
  
#include <libio/io.h>  
  
ssize\_t print\_fd(const int fd, char \*fmt, ...) {  
 va\_list args;  
 va\_start(args, fmt);  
 char buff[IO\_MAX\_STR\_LEN];  
 size\_t len = vsnprintf(buff, IO\_MAX\_STR\_LEN - 1, fmt, args);  
 const ssize\_t writen\_bytes = write(fd, buff, len);  
 va\_end(args);  
 return writen\_bytes;  
}  
ssize\_t write\_str(const int fd, const char \*buff) {  
 return write(fd, buff, strlen(buff));  
}  
  
ssize\_t reads\_fd(const int fd, char \*buff, const size\_t buff\_size) {  
 ssize\_t read\_bytes = 0;  
 return read(fd, buff, buff\_size);  
}

**child.c**

*/\*\*  
 \* @file  
 \* @brief  
 \* @details  
 \* @author xsestech   
 \* @date 26.10.2024  
 \*/*#include <stdio.h>  
#include <stdlib.h>  
#include <unistd.h>  
#include <fcntl.h>  
#include <libconfig/config.h>  
  
#include <libio/io.h>  
#include <libio/shared/shared.h>  
  
int main(const int argc, char \*argv[]) {  
 if (argc != 2) {  
 print\_fd(STDERR\_FILENO, "No file specified");  
 exit(EXIT\_FAILURE);  
 }  
 const pid\_t pid = getpid();  
 int file = open(argv[1], O\_WRONLY | O\_CREAT | O\_TRUNC | O\_APPEND, 0600);  
  
 if (file == -1) {  
 print\_fd(STDERR\_FILENO, "%d: Error opening file %s\n", pid, argv[1]);  
 exit(EXIT\_FAILURE);  
 }  
  
 shared\_handle\_t shared = shared\_consumer\_open(argv[0], 4096);  
 if (shared == NULL) {  
 print\_fd(STDERR\_FILENO, "%d: Error accessing shared memory\n", pid);  
 exit(EXIT\_FAILURE);  
 }  
  
 print\_fd(STDOUT\_FILENO, "%d: opened file %s\n", getpid(), argv[1]);  
 char\* str;;  
 ssize\_t bytes = 0;  
 while ((str = shared\_get\_str(shared)) != NULL) {  
  
 if (str[0] == '\n') {  
 break;  
 }  
  
  
 str[strlen(str) - 1] = '\0'; // remove newline  
 print\_fd(STDOUT\_FILENO, "%d: got: %s\n", pid, str);  
 if (print\_fd(file, "%s ", str) == -1) {  
 print\_fd(STDERR\_FILENO, "%d: Error writing to file\n", pid);  
 exit(EXIT\_FAILURE);  
 }  
 }  
 const char term = '\0';  
 write(file, &term, sizeof(term));  
 close(file);  
 shared\_close(shared);  
 return 0;  
}

**shared.c**

*/\*\*  
 \* @file  
 \* @brief  
 \* @details  
 \* @author xsestech   
 \* @date 12.12.2024  
 \*/*#include "shared.h"  
#include <libio/io.h>  
#include <errno.h>  
shared\_handle\_t shared\_open(char \*name, size\_t size, bool is\_producer) {  
 if (name == NULL || strlen(name) > SHARED\_MAX\_NAME\_LEN) {  
 return NULL;  
 }  
 int rw\_perms = SHARED\_READ\_PERMISSIONS;  
 if (is\_producer) rw\_perms = SHARED\_WRITE\_PERMISSIONS;  
 int fd = shm\_open(name, rw\_perms, SHARED\_ACCESS\_PERMISSIONS);  
 if (fd < 0) {  
 print\_fd(STDERR\_FILENO, "file descr\n");  
 return NULL;  
 }  
 ftruncate(fd, size);  
  
 char \*mem\_ptr = mmap(0, size, SHARED\_DEFAULT\_PROTECTIONS, MAP\_SHARED, fd, 0);  
 if ((caddr\_t) mem\_ptr == (caddr\_t) -1) {  
 print\_fd(STDERR\_FILENO, " mmap errno: %s \n", strerror(errno));  
 close(fd);  
 return NULL;  
 }  
  
 char data\_ready\_name[SHARED\_MAX\_NAME\_LEN];  
 char buffer\_empty\_name[SHARED\_MAX\_NAME\_LEN];  
 snprintf(data\_ready\_name, sizeof(data\_ready\_name), "%s\_data", name);  
 snprintf(buffer\_empty\_name, sizeof(buffer\_empty\_name), "%s\_buff", name);  
  
 sem\_t \*data\_ready = sem\_open(data\_ready\_name, O\_CREAT, SHARED\_ACCESS\_PERMISSIONS, 0);  
 if (data\_ready == SEM\_FAILED) {  
 print\_fd(STDERR\_FILENO, " sem \n");  
 close(fd);  
 munmap(mem\_ptr, size);  
 return NULL;  
 }  
 sem\_t \*buffer\_empty = sem\_open(buffer\_empty\_name, O\_CREAT, SHARED\_ACCESS\_PERMISSIONS, 1);  
 if (buffer\_empty == SEM\_FAILED) {  
 sem\_close(data\_ready);  
 print\_fd(STDERR\_FILENO, " sem \n");  
 close(fd);  
 munmap(mem\_ptr, size);  
 return NULL;  
 }  
  
 shared\_handle\_t shared = malloc(sizeof(shared\_t));  
 assert(shared);  
 strcpy(shared->name, name);  
 shared->is\_producer = is\_producer;  
 shared->size = size;  
 shared->fd = fd;  
 shared->mem\_ptr = mem\_ptr;  
 shared->buffer\_empty = buffer\_empty;  
 shared->data\_ready = data\_ready;  
 return shared;  
}  
shared\_handle\_t shared\_producer\_open(char \*name, size\_t size) {  
 return shared\_open(name, size, true);  
}  
shared\_handle\_t shared\_consumer\_open(char \*name, size\_t size) {  
 return shared\_open(name, size, false);  
}  
char \*shared\_get\_str(shared\_handle\_t shared) {  
 if (sem\_wait(shared->data\_ready) != 0) {  
 return NULL;  
 }  
  
 char \*str = strdup(shared->mem\_ptr);  
 if (!str) {  
 sem\_post(shared->data\_ready);  
 return NULL;  
 }  
 if (sem\_post(shared->buffer\_empty) != 0) {  
 free(str);  
 return NULL;  
 }  
 return str;  
}  
  
int shared\_put\_str(shared\_handle\_t shared, char \*str) {  
 if (strlen(str) > shared->size) return -1;  
 if (sem\_wait(shared->buffer\_empty) != 0) {  
 return -1;  
 }  
 strcpy(shared->mem\_ptr, str);  
 return sem\_post(shared->data\_ready);  
}  
  
void shared\_close(shared\_handle\_t shared) {  
 munmap(shared->mem\_ptr, shared->size);  
 close(shared->fd);  
 sem\_close(shared->data\_ready);  
 sem\_close(shared->buffer\_empty);  
 unlink(shared->name);  
 free(shared);  
}

**Протокол работы программы**

**Тестирование:**

builder@4c1c3dd98286:/app$ ./build/parent/parent

file1.out

file2.out

Parent 1: created child with pid 398

build/child/child: child pid 398

Parent 1: created child with pid 399

build/child/child: child pid 399

398: opened file file1.out

399: opened file file2.out

123

399: got: 123

234

398: got: 234

123

399: got: 123

234

398: got: 234

123

399: got: 123

234

398: got: 234

Child 398: exit status 0

Child 399: exit status 0

cat file1.out

234234234

$ cat < file2.out

123123123

**Strace:**

builder@4c1c3dd98286:/app$ strace -f ./build/parent/parent

execve("./cmake-build-linux/parent/parent", ["./cmake-build-linux/parent/paren"...], 0xffffd56e3348 /\* 14 vars \*/) = 0

brk(NULL) = 0xaaaade2bc000

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0xffff87ecc000

faccessat(AT\_FDCWD, "/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=13739, ...}) = 0

mmap(NULL, 13739, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0xffff87ec8000

close(3) = 0

openat(AT\_FDCWD, "/lib/aarch64-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\267\0\1\0\0\0\360\206\2\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=1722920, ...}) = 0

mmap(NULL, 1892240, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_DENYWRITE, -1, 0) = 0xffff87cc5000

mmap(0xffff87cd0000, 1826704, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0) = 0xffff87cd0000

munmap(0xffff87cc5000, 45056) = 0

munmap(0xffff87e8e000, 20368) = 0

mprotect(0xffff87e6a000, 77824, PROT\_NONE) = 0

mmap(0xffff87e7d000, 20480, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x19d000) = 0xffff87e7d000

mmap(0xffff87e82000, 49040, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0xffff87e82000

close(3) = 0

set\_tid\_address(0xffff87eccf90) = 82

set\_robust\_list(0xffff87eccfa0, 24) = 0

rseq(0xffff87ecd5e0, 0x20, 0, 0xd428bc00) = 0

mprotect(0xffff87e7d000, 12288, PROT\_READ) = 0

mprotect(0xaaaad73af000, 4096, PROT\_READ) = 0

mprotect(0xffff87ed1000, 8192, PROT\_READ) = 0

prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

munmap(0xffff87ec8000, 13739) = 0

**read**(0, test/file1.out

"test/file1.out\n", 4096) = 15

**read**(0, test/file2.out

"test/file2.out\n", 4096) = 15

**openat**(AT\_FDCWD, "/dev/shm/child1", O\_RDWR|O\_CREAT|O\_NOFOLLOW|O\_CLOEXEC, 0666) = 3

**ftruncate**(3, 1024) = 0

**mmap**(NULL, 1024, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 3, 0) = 0xffff87ecb000

**openat**(AT\_FDCWD, "/dev/shm/sem.child1\_data", O\_RDWR|O\_NOFOLLOW|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

getrandom("\xd4\xbb\x2a\x4e\x2f\x5e\x38\x6b", 8, GRND\_NONBLOCK) = 8

newfstatat(AT\_FDCWD, "/dev/shm/sem.qIOtqm", 0xffffc8b2bc38, AT\_SYMLINK\_NOFOLLOW) = -1 ENOENT (No such file or directory)

**openat**(AT\_FDCWD, "/dev/shm/sem.qIOtqm", O\_RDWR|O\_CREAT|O\_EXCL|O\_NOFOLLOW|O\_CLOEXEC, 0666) = 4

**write**(4, "\0\0\0\0\0\0\0\0\200\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 32) = 32

**mmap**(NULL, 32, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 4, 0) = 0xffff87eca000

linkat(AT\_FDCWD, "/dev/shm/sem.qIOtqm", AT\_FDCWD, "/dev/shm/sem.child1\_data", 0) = 0

fstat(4, {st\_mode=S\_IFREG|0644, st\_size=32, ...}) = 0

getrandom("\x81\xda\x08\x36\xd9\x8c\xae\x46", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0xaaaade2bc000

brk(0xaaaade2dd000) = 0xaaaade2dd000

unlinkat(AT\_FDCWD, "/dev/shm/sem.qIOtqm", 0) = 0

close(4) = 0

**openat**(AT\_FDCWD, "/dev/shm/sem.child1\_buff", O\_RDWR|O\_NOFOLLOW|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

getrandom("\x1a\xa6\xf6\xbc\xa7\x63\xe0\x85", 8, GRND\_NONBLOCK) = 8

newfstatat(AT\_FDCWD, "/dev/shm/sem.wqhT7y", 0xffffc8b2bc38, AT\_SYMLINK\_NOFOLLOW) = -1 ENOENT (No such file or directory)

**openat**(AT\_FDCWD, "/dev/shm/sem.wqhT7y", O\_RDWR|O\_CREAT|O\_EXCL|O\_NOFOLLOW|O\_CLOEXEC, 0666) = 4

write(4, "\1\0\0\0\0\0\0\0\200\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 32) = 32

mmap(NULL, 32, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 4, 0) = 0xffff87ec9000

linkat(AT\_FDCWD, "/dev/shm/sem.wqhT7y", AT\_FDCWD, "/dev/shm/sem.child1\_buff", 0) = 0

fstat(4, {st\_mode=S\_IFREG|0644, st\_size=32, ...}) = 0

unlinkat(AT\_FDCWD, "/dev/shm/sem.wqhT7y", 0) = 0

close(4) = 0

**clone**(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLDstrace: Process 83 attached

, child\_tidptr=0xffff87eccf90) = 83

[pid 83] set\_robust\_list(0xffff87eccfa0, 24 <unfinished ...>

[pid 82] **getppid**( <unfinished ...>

[pid 83] <... set\_robust\_list resumed>) = 0

[pid 82] <... getppid resumed>) = 79

[pid 83] getpid( <unfinished ...>

[pid 82] **write**(1, "Parent 79: created child with pi"..., 37 <unfinished ...>

[pid 83] <... getpid resumed>) = 83

Parent 79: created child with pid 83

[pid 82] <... write resumed>) = 37

[pid 83] fstat(1, <unfinished ...>

[pid 82] **openat**(AT\_FDCWD, "/dev/shm/child2", O\_RDWR|O\_CREAT|O\_NOFOLLOW|O\_CLOEXEC, 0666 <unfinished ...>

[pid 83] <... fstat resumed>{st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}) = 0

[pid 82] <... openat resumed>) = 4

[pid 83] **write**(1, "cmake-build-linux/child/child: c"..., 44cmake-build-linux/child/child: child pid 83

<unfinished ...>

[pid 82] **ftruncate**(4, 1024 <unfinished ...>

[pid 83] <... write resumed>) = 44

[pid 82] <... ftruncate resumed>) = 0

[pid 83] **execve**("cmake-build-linux/child/child", ["child1", "test/file1.out"], 0xffffc8b327b8 /\* 14 vars \*/ <unfinished ...>

[pid 82] mmap(NULL, 1024, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 4, 0) = 0xffff87ec8000

[pid 82] openat(AT\_FDCWD, "/dev/shm/sem.child2\_data", O\_RDWR|O\_NOFOLLOW|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

[pid 82] getrandom("\x0c\xb2\x66\xb5\x4d\xf7\x94\xfe", 8, GRND\_NONBLOCK) = 8

[pid 82] getrandom("\xd0\x54\x8e\xe4\x30\x36\xe1\xe6", 8, GRND\_NONBLOCK) = 8

[pid 82] newfstatat(AT\_FDCWD, "/dev/shm/sem.cLIGJ7", 0xffffc8b2bc38, AT\_SYMLINK\_NOFOLLOW) = -1 ENOENT (No such file or directory)

[pid 82] openat(AT\_FDCWD, "/dev/shm/sem.cLIGJ7", O\_RDWR|O\_CREAT|O\_EXCL|O\_NOFOLLOW|O\_CLOEXEC, 0666) = 5

[pid 82] write(5, "\0\0\0\0\0\0\0\0\200\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 32) = 32

[pid 82] **mmap**(NULL, 32, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 5, 0) = 0xffff87ec7000

[pid 82] linkat(AT\_FDCWD, "/dev/shm/sem.cLIGJ7", AT\_FDCWD, "/dev/shm/sem.child2\_data", 0) = 0

[pid 82] fstat(5, <unfinished ...>

[pid 83] <... execve resumed>) = 0

[pid 82] <... fstat resumed>{st\_mode=S\_IFREG|0644, st\_size=32, ...}) = 0

[pid 83] brk(NULL <unfinished ...>

[pid 82] unlinkat(AT\_FDCWD, "/dev/shm/sem.cLIGJ7", 0 <unfinished ...>

[pid 83] <... brk resumed>) = 0xaaaad30eb000

[pid 82] <... unlinkat resumed>) = 0

[pid 83] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

[pid 82] close(5 <unfinished ...>

[pid 83] <... mmap resumed>) = 0xffffb5176000

[pid 82] <... close resumed>) = 0

[pid 83] faccessat(AT\_FDCWD, "/etc/ld.so.preload", R\_OK <unfinished ...>

[pid 82] openat(AT\_FDCWD, "/dev/shm/sem.child2\_buff", O\_RDWR|O\_NOFOLLOW|O\_CLOEXEC <unfinished ...>

[pid 83] <... faccessat resumed>) = -1 ENOENT (No such file or directory)

[pid 82] <... openat resumed>) = -1 ENOENT (No such file or directory)

[pid 83] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC <unfinished ...>

[pid 82] getrandom( <unfinished ...>

[pid 83] <... openat resumed>) = 3

[pid 82] <... getrandom resumed>"\xb1\x44\xe1\xb5\xc9\x42\x00\x37", 8, GRND\_NONBLOCK) = 8

[pid 83] fstat(3, <unfinished ...>

[pid 82] newfstatat(AT\_FDCWD, "/dev/shm/sem.zA1jQl", <unfinished ...>

[pid 83] <... fstat resumed>{st\_mode=S\_IFREG|0644, st\_size=13739, ...}) = 0

[pid 82] <... newfstatat resumed>0xffffc8b2bc38, AT\_SYMLINK\_NOFOLLOW) = -1 ENOENT (No such file or directory)

[pid 83] mmap(NULL, 13739, PROT\_READ, MAP\_PRIVATE, 3, 0 <unfinished ...>

[pid 82] **openat**(AT\_FDCWD, "/dev/shm/sem.zA1jQl", O\_RDWR|O\_CREAT|O\_EXCL|O\_NOFOLLOW|O\_CLOEXEC, 0666) = 5

[pid 83] <... mmap resumed>) = 0xffffb5172000

[pid 82] **write**(5, "\1\0\0\0\0\0\0\0\200\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 32 <unfinished ...>

[pid 83] close(3 <unfinished ...>

[pid 82] <... write resumed>) = 32

[pid 83] <... close resumed>) = 0

[pid 82] mmap(NULL, 32, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 5, 0 <unfinished ...>

[pid 83] openat(AT\_FDCWD, "/lib/aarch64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC <unfinished ...>

[pid 82] <... mmap resumed>) = 0xffff87ec6000

[pid 83] <... openat resumed>) = 3

[pid 82] linkat(AT\_FDCWD, "/dev/shm/sem.zA1jQl", AT\_FDCWD, "/dev/shm/sem.child2\_buff", 0 <unfinished ...>

[pid 83] read(3, <unfinished ...>

[pid 82] <... linkat resumed>) = 0

[pid 82] fstat(5, <unfinished ...>

[pid 83] <... read resumed>"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0\267\0\1\0\0\0\360\206\2\0\0\0\0\0"..., 832) = 832

[pid 82] <... fstat resumed>{st\_mode=S\_IFREG|0644, st\_size=32, ...}) = 0

[pid 83] fstat(3, <unfinished ...>

[pid 82] unlinkat(AT\_FDCWD, "/dev/shm/sem.zA1jQl", 0 <unfinished ...>

[pid 83] <... fstat resumed>{st\_mode=S\_IFREG|0755, st\_size=1722920, ...}) = 0

[pid 82] <... unlinkat resumed>) = 0

[pid 83] mmap(NULL, 1892240, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_DENYWRITE, -1, 0 <unfinished ...>

[pid 82] close(5 <unfinished ...>

[pid 83] <... mmap resumed>) = 0xffffb4f6f000

[pid 82] <... close resumed>) = 0

[pid 83] mmap(0xffffb4f70000, 1826704, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0 <unfinished ...>

[pid 82] **clone**(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD <unfinished ...>

[pid 83] <... mmap resumed>) = 0xffffb4f70000

[pid 83] munmap(0xffffb4f6f000, 4096strace: Process 84 attached

) = 0

[pid 82] <... clone resumed>, child\_tidptr=0xffff87eccf90) = 84

[pid 84] set\_robust\_list(0xffff87eccfa0, 24 <unfinished ...>

[pid 83] munmap(0xffffb512e000, 61328 <unfinished ...>

[pid 82] getppid( <unfinished ...>

[pid 84] <... set\_robust\_list resumed>) = 0

[pid 83] <... munmap resumed>) = 0

[pid 82] <... getppid resumed>) = 79

[pid 84] **getpid**( <unfinished ...>

[pid 83] mprotect(0xffffb510a000, 77824, PROT\_NONE <unfinished ...>

[pid 82] **write**(1, "Parent 79: created child with pi"..., 37 <unfinished ...>

[pid 84] <... getpid resumed>) = 84

[pid 83] <... mprotect resumed>) = 0

Parent 79: created child with pid 84

[pid 82] <... write resumed>) = 37

[pid 83] mmap(0xffffb511d000, 20480, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x19d000 <unfinished ...>

[pid 84] fstat(1, <unfinished ...>

[pid 82] read(0, <unfinished ...>

[pid 84] <... fstat resumed>{st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}) = 0

[pid 83] <... mmap resumed>) = 0xffffb511d000

[pid 84] write(1, "cmake-build-linux/child/child: c"..., 44 <unfinished ...>

[pid 83] mmap(0xffffb5122000, 49040, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0cmake-build-linux/child/child: child pid 84

<unfinished ...>

[pid 84] <... write resumed>) = 44

[pid 83] <... mmap resumed>) = 0xffffb5122000

[pid 84] **execve**("cmake-build-linux/child/child", ["child2", "test/file2.out"], 0xffffc8b327b8 /\* 14 vars \*/ <unfinished ...>

[pid 83] close(3) = 0

[pid 83] set\_tid\_address(0xffffb5176f90) = 83

[pid 83] set\_robust\_list(0xffffb5176fa0, 24) = 0

[pid 83] rseq(0xffffb51775e0, 0x20, 0, 0xd428bc00) = 0

[pid 83] mprotect(0xffffb511d000, 12288, PROT\_READ) = 0

[pid 84] <... execve resumed>) = 0

[pid 83] mprotect(0xaaaab82cf000, 4096, PROT\_READ <unfinished ...>

[pid 84] brk(NULL <unfinished ...>

[pid 83] <... mprotect resumed>) = 0

[pid 84] <... brk resumed>) = 0xaaaac45ec000

[pid 83] mprotect(0xffffb517b000, 8192, PROT\_READ <unfinished ...>

[pid 84] **mmap**(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

[pid 83] <... mprotect resumed>) = 0

[pid 84] <... mmap resumed>) = 0xffff9aa63000

[pid 83] prlimit64(0, RLIMIT\_STACK, NULL, <unfinished ...>

[pid 84] faccessat(AT\_FDCWD, "/etc/ld.so.preload", R\_OK <unfinished ...>

[pid 83] <... prlimit64 resumed>{rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

[pid 84] <... faccessat resumed>) = -1 ENOENT (No such file or directory)

[pid 83] munmap(0xffffb5172000, 13739 <unfinished ...>

[pid 84] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC <unfinished ...>

[pid 83] <... munmap resumed>) = 0

[pid 84] <... openat resumed>) = 3

[pid 83] getpid( <unfinished ...>

[pid 84] fstat(3, <unfinished ...>

[pid 83] <... getpid resumed>) = 83

[pid 84] <... fstat resumed>{st\_mode=S\_IFREG|0644, st\_size=13739, ...}) = 0

[pid 83] openat(AT\_FDCWD, "test/file1.out", O\_WRONLY|O\_CREAT|O\_TRUNC|O\_APPEND, 0600 <unfinished ...>

[pid 84] mmap(NULL, 13739, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0xffff9aa5f000

[pid 84] close(3) = 0

[pid 84] openat(AT\_FDCWD, "/lib/aarch64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

[pid 84] read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0\267\0\1\0\0\0\360\206\2\0\0\0\0\0"..., 832) = 832

[pid 84] fstat(3, {st\_mode=S\_IFREG|0755, st\_size=1722920, ...}) = 0

[pid 84] mmap(NULL, 1892240, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_DENYWRITE, -1, 0) = 0xffff9a85c000

[pid 84] mmap(0xffff9a860000, 1826704, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0) = 0xffff9a860000

[pid 83] <... openat resumed>) = 3

[pid 84] munmap(0xffff9a85c000, 16384) = 0

[pid 83] **openat**(AT\_FDCWD, "/dev/shm/child1", O\_RDWR|O\_NOFOLLOW|O\_CLOEXEC <unfinished ...>

[pid 84] munmap(0xffff9aa1e000, 49040 <unfinished ...>

[pid 83] <... openat resumed>) = 4

[pid 84] <... munmap resumed>) = 0

[pid 83] **ftruncate**(4, 4096 <unfinished ...>

[pid 84] mprotect(0xffff9a9fa000, 77824, PROT\_NONE <unfinished ...>

[pid 83] <... ftruncate resumed>) = 0

[pid 84] <... mprotect resumed>) = 0

[pid 84] mmap(0xffff9aa0d000, 20480, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x19d000) = 0xffff9aa0d000

[pid 83] mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 4, 0 <unfinished ...>

[pid 84] mmap(0xffff9aa12000, 49040, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

[pid 83] <... mmap resumed>) = 0xffffb5175000

[pid 84] <... mmap resumed>) = 0xffff9aa12000

[pid 83] openat(AT\_FDCWD, "/dev/shm/sem.child1\_data", O\_RDWR|O\_NOFOLLOW|O\_CLOEXEC <unfinished ...>

[pid 84] close(3 <unfinished ...>

[pid 83] <... openat resumed>) = 5

[pid 84] <... close resumed>) = 0

[pid 83] fstat(5, <unfinished ...>

[pid 84] set\_tid\_address(0xffff9aa63f90 <unfinished ...>

[pid 83] <... fstat resumed>{st\_mode=S\_IFREG|0644, st\_size=32, ...}) = 0

[pid 84] <... set\_tid\_address resumed>) = 84

[pid 83] getrandom( <unfinished ...>

[pid 84] set\_robust\_list(0xffff9aa63fa0, 24 <unfinished ...>

[pid 83] <... getrandom resumed>"\x1d\x1d\x9d\x0e\xf1\xd8\xe7\x22", 8, GRND\_NONBLOCK) = 8

[pid 84] <... set\_robust\_list resumed>) = 0

[pid 84] rseq(0xffff9aa645e0, 0x20, 0, 0xd428bc00 <unfinished ...>

[pid 83] brk(NULL <unfinished ...>

[pid 84] <... rseq resumed>) = 0

[pid 83] <... brk resumed>) = 0xaaaad30eb000

[pid 84] mprotect(0xffff9aa0d000, 12288, PROT\_READ <unfinished ...>

[pid 83] brk(0xaaaad310c000 <unfinished ...>

[pid 84] <... mprotect resumed>) = 0

[pid 83] <... brk resumed>) = 0xaaaad310c000

[pid 84] mprotect(0xaaaab2d7f000, 4096, PROT\_READ <unfinished ...>

[pid 83] mmap(NULL, 32, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 5, 0 <unfinished ...>

[pid 84] <... mprotect resumed>) = 0

[pid 83] <... mmap resumed>) = 0xffffb5174000

[pid 84] mprotect(0xffff9aa68000, 8192, PROT\_READ <unfinished ...>

[pid 83] close(5 <unfinished ...>

[pid 84] <... mprotect resumed>) = 0

[pid 83] <... close resumed>) = 0

[pid 84] prlimit64(0, RLIMIT\_STACK, NULL, <unfinished ...>

[pid 83] openat(AT\_FDCWD, "/dev/shm/sem.child1\_buff", O\_RDWR|O\_NOFOLLOW|O\_CLOEXEC <unfinished ...>

[pid 84] <... prlimit64 resumed>{rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

[pid 83] <... openat resumed>) = 5

[pid 84] munmap(0xffff9aa5f000, 13739 <unfinished ...>

[pid 83] fstat(5, <unfinished ...>

[pid 84] <... munmap resumed>) = 0

[pid 83] <... fstat resumed>{st\_mode=S\_IFREG|0644, st\_size=32, ...}) = 0

[pid 84] **getpid**( <unfinished ...>

[pid 83] mmap(NULL, 32, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 5, 0 <unfinished ...>

[pid 84] <... getpid resumed>) = 84

[pid 83] <... mmap resumed>) = 0xffffb5173000

[pid 84] openat(AT\_FDCWD, "test/file2.out", O\_WRONLY|O\_CREAT|O\_TRUNC|O\_APPEND, 0600 <unfinished ...>

[pid 83] close(5) = 0

[pid 83] getpid() = 83

[pid 83] write(1, "83: opened file test/file1.out\n", 3183: opened file test/file1.out

) = 31

[pid 83] futex(0xffffb5174000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 84] <... openat resumed>) = 3

[pid 84] openat(AT\_FDCWD, "/dev/shm/child2", O\_RDWR|O\_NOFOLLOW|O\_CLOEXEC) = 4

[pid 84] ftruncate(4, 4096) = 0

[pid 84] mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 4, 0) = 0xffff9aa62000

[pid 84] openat(AT\_FDCWD, "/dev/shm/sem.child2\_data", O\_RDWR|O\_NOFOLLOW|O\_CLOEXEC) = 5

[pid 84] fstat(5, {st\_mode=S\_IFREG|0644, st\_size=32, ...}) = 0

[pid 84] getrandom("\x5b\xa8\xdb\xa2\xb9\x5a\x3b\xa3", 8, GRND\_NONBLOCK) = 8

[pid 84] brk(NULL) = 0xaaaac45ec000

[pid 84] brk(0xaaaac460d000) = 0xaaaac460d000

[pid 84] mmap(NULL, 32, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 5, 0) = 0xffff9aa61000

[pid 84] close(5) = 0

[pid 84] openat(AT\_FDCWD, "/dev/shm/sem.child2\_buff", O\_RDWR|O\_NOFOLLOW|O\_CLOEXEC) = 5

[pid 84] fstat(5, {st\_mode=S\_IFREG|0644, st\_size=32, ...}) = 0

[pid 84] mmap(NULL, 32, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 5, 0) = 0xffff9aa60000

[pid 84] close(5) = 0

[pid 84] getpid() = 84

[pid 84] **write**(1, "84: opened file test/file2.out\n", 3184: opened file test/file2.out

) = 31

[pid 84] futex(0xffff9aa61000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY123

<unfinished ...>

[pid 82] <... read resumed>"123\n", 1024) = 4

[pid 82] futex(0xffff87ec7000, FUTEX\_WAKE, 1) = 1

[pid 84] <... futex resumed>) = 0

[pid 82] read(0, <unfinished ...>

[pid 84] write(1, "84: got: 123\n", 1384: got: 123

) = 13

[pid 84] **write**(3, "123 ", 4) = 4

[pid 84] futex(0xffff9aa61000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY234

<unfinished ...>

[pid 82] <... read resumed>"234\n", 1024) = 4

[pid 82] futex(0xffff87eca000, FUTEX\_WAKE, 1) = 1

[pid 83] <... futex resumed>) = 0

[pid 82] read(0, <unfinished ...>

[pid 83] write(1, "83: got: 234\n", 1383: got: 234

) = 13

[pid 83] write(3, "234 ", 4) = 4

[pid 83] futex(0xffffb5174000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY123

<unfinished ...>

[pid 82] <... read resumed>"123\n", 1024) = 4

[pid 82] futex(0xffff87ec7000, FUTEX\_WAKE, 1) = 1

[pid 84] <... futex resumed>) = 0

[pid 82] read(0, <unfinished ...>

[pid 84] write(1, "84: got: 123\n", 1384: got: 123

) = 13

[pid 84] **write**(3, "123 ", 4) = 4

[pid 84] futex(0xffff9aa61000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY234

<unfinished ...>

[pid 82] <... read resumed>"234\n", 1024) = 4

[pid 82] futex(0xffff87eca000, FUTEX\_WAKE, 1) = 1

[pid 83] <... futex resumed>) = 0

[pid 82] read(0, <unfinished ...>

[pid 83] write(1, "83: got: 234\n", 1383: got: 234

) = 13

[pid 83] **write**(3, "234 ", 4) = 4

[pid 83] futex(0xffffb5174000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY123

<unfinished ...>

[pid 82] <... read resumed>"123\n", 1024) = 4

[pid 82] futex(0xffff87ec7000, FUTEX\_WAKE, 1) = 1

[pid 84] <... futex resumed>) = 0

[pid 82] **read**(0, <unfinished ...>

[pid 84] **write**(1, "84: got: 123\n", 1384: got: 123

) = 13

[pid 84] **write**(3, "123 ", 4) = 4

[pid 84] futex(0xffff9aa61000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY324

<unfinished ...>

[pid 82] <... read resumed>"324\n", 1024) = 4

[pid 82] futex(0xffff87eca000, FUTEX\_WAKE, 1) = 1

[pid 82] **read**(0, <unfinished ...>

[pid 83] <... futex resumed>) = 0

[pid 83] **write**(1, "83: got: 324\n", 1383: got: 324

) = 13

[pid 83] **write**(3, "324 ", 4) = 4

[pid 83] futex(0xffffb5174000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY

<unfinished ...>

[pid 82] <... read resumed>"\n", 1024) = 1

[pid 82] futex(0xffff87eca000, FUTEX\_WAKE, 1) = 1

[pid 83] <... futex resumed>) = 0

[pid 82] futex(0xffff87ec7000, FUTEX\_WAKE, 1 <unfinished ...>

[pid 83] **write**(3, "\0", 1 <unfinished ...>

[pid 82] <... futex resumed>) = 1

[pid 84] <... futex resumed>) = 0

[pid 82] munmap(0xffff87ecb000, 1024 <unfinished ...>

[pid 84] write(3, "\0", 1 <unfinished ...>

[pid 82] <... munmap resumed>) = 0

[pid 82] close(3) = 0

[pid 82] munmap(0xffff87eca000, 32) = 0

[pid 83] <... write resumed>) = 1

[pid 82] munmap(0xffff87ec9000, 32 <unfinished ...>

[pid 83] close(3 <unfinished ...>

[pid 84] <... write resumed>) = 1

[pid 84] close(3 <unfinished ...>

[pid 82] <... munmap resumed>) = 0

[pid 82] **unlinkat**(AT\_FDCWD, "child1", 0 <unfinished ...>

[pid 84] <... close resumed>) = 0

[pid 83] <... close resumed>) = 0

[pid 84] **munmap**(0xffff9aa62000, 4096 <unfinished ...>

[pid 83] **munmap**(0xffffb5175000, 4096 <unfinished ...>

[pid 82] <... unlinkat resumed>) = -1 ENOENT (No such file or directory)

[pid 84] <... munmap resumed>) = 0

[pid 83] <... munmap resumed>) = 0

[pid 82] **wait4**(83, <unfinished ...>

[pid 84] close(4 <unfinished ...>

[pid 83] close(4 <unfinished ...>

[pid 84] <... close resumed>) = 0

[pid 83] <... close resumed>) = 0

[pid 84] munmap(0xffff9aa61000, 32 <unfinished ...>

[pid 83] munmap(0xffffb5174000, 32 <unfinished ...>

[pid 84] <... munmap resumed>) = 0

[pid 83] <... munmap resumed>) = 0

[pid 84] munmap(0xffff9aa60000, 32 <unfinished ...>

[pid 83] munmap(0xffffb5173000, 32) = 0

[pid 83] **unlinkat**(AT\_FDCWD, "child1", 0 <unfinished ...>

[pid 84] <... munmap resumed>) = 0

[pid 84] **unlinkat**(AT\_FDCWD, "child2", 0 <unfinished ...>

[pid 83] <... unlinkat resumed>) = -1 ENOENT (No such file or directory)

[pid 83] exit\_group(0 <unfinished ...>

[pid 84] <... unlinkat resumed>) = -1 ENOENT (No such file or directory)

[pid 83] <... exit\_group resumed>) = ?

[pid 84] exit\_group(0) = ?

[pid 83] +++ exited with 0 +++

[pid 82] <... wait4 resumed>[{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 83

[pid 84] +++ exited with 0 +++

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=83, si\_uid=501, si\_status=0, si\_utime=0, si\_stime=0} ---

write(1, "Child 83: exit status 0\n", 24Child 83: exit status 0

) = 24

munmap(0xffff87ec8000, 1024) = 0

close(4) = 0

munmap(0xffff87ec7000, 32) = 0

munmap(0xffff87ec6000, 32) = 0

unlinkat(AT\_FDCWD, "child2", 0) = -1 ENOENT (No such file or directory)

wait4(84, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 84

write(1, "Child 84: exit status 0\n", 24Child 84: exit status 0

) = 24

exit\_group(0) = ?

+++ exited with 0 +++

**Вывод**

В ходе данной работы я научился создавать процессы, налаживать общение между ними с помощью shared memory. Столкнулся с проблемами при синхронизации с помощью семафоров.