МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №5
по курсу «Операционные системы»

Выполнил: А. А. Устинов
Группа: М8О-207БВ-24
Преподаватель: Е. С. Миронов

Москва, 2025

Условие

Цель работы: Приобретение практических навыков диагностики работы программного обеспечения.

Задание: При выполнении лабораторных работ по курсу ОС необходимо продемонстрировать ключевые системные вызовы, которые в них используются и то, что их использование соответствует варианту ЛР.

Вариант: -

Метод решения

Для проверки корректности выполненных лабораторных работ, а также для подтверждения использования предписанных системных вызовов, все разработанные программы были проанализированы с применением утилиты `strace`. Данная утилита предоставляет возможность отслеживать выполнение системных вызовов в процессе работы программы, фиксируя как передаваемые параметры, так и возвращаемые результаты.

Для каждой лабораторной работы формировался отдельный файл трассировки посредством команды следующего вида:
`strace -o trace_labN.log ./labN_executable [аргументы]`,
где `trace_labN.log` — файл с результатами трассировки, а `labN_executable` — исполняемый файл, реализующий задание лабораторной работы номер `N`.

Полученные логи подвергались детальному анализу, включающему ручную проверку и элементы автоматизированной обработки, с целью:

- идентификации системных вызовов, обязательных в рамках соответствующего задания;
- контроля корректности их применения, включая последовательность вызовов, корректную обработку ошибок и допустимость передаваемых аргументов;
- подтверждения соответствия фактического поведения программы предполагаемой логике выполнения, включая управление процессами, механизмы синхронизации и обмен данными.

Описание Программы

Анализ системных вызовов лабораторной работы 1

1. `execve("./parent ["./parent"], 0x7fff77b42d10) = 0`

   Системный вызов выполняет загрузку и запуск исполняемого файла, полностью заменяя текущий образ процесса.

   - `"./parent"` — путь к исполняемому файлу;
   - `["./parent"]` — массив аргументов командной строки (`argv`);
   - `0x7fff77b42d10` — указатель на массив переменных окружения (`envp`).

2. `pipe2([3, 4], 0) = 0`

   Создаётся неименованный канал для межпроцессного взаимодействия.

- 3 — файловый дескриптор конца канала для чтения;
- 4 — файловый дескриптор конца канала для записи;
- 0 — отсутствие флагов (канал создаётся без дополнительных опций).

3. `write(1, "Filename:\n 10) = 10`

   Выполняется вывод строки-приглашения пользователю.

   - 1 — стандартный поток вывода (stdout);
   - "Filename:\n" — выводимая строка;
   - 10 — количество записываемых байт.

4. `read(0, "input.txt\n 1024) = 10`

   Осуществляется чтение имени файла из стандартного потока ввода.

   - 0 — стандартный поток ввода (stdin);
   - буфер для сохранения считанных данных;
   - 1024 — максимальное число байт для чтения.

5. `openat(AT_FDCWD, "input.txt O_RDONLY) = 5`

   Открывается файл для последующего чтения.

   - AT_FDCWD — текущий рабочий каталог;
   - "input.txt" — имя файла;
   - O_RDONLY — режим открытия «только для чтения».

   Файлу присвоен дескриптор 5.

6. `clone(...) = 6474`

   Создаётся дочерний процесс. Аргументы скрыты в трассировке, но включают флаги клонирования и параметры стека. Возвращаемое значение 6474 соответствует идентификатору дочернего процесса.

7. `read(3, "Sum: 3354.30\n 4096) = 13`

   Родительский процесс считывает данные из канала.

   - 3 — файловый дескриптор чтения из pipe;
   - буфер для принимаемых данных;
   - 4096 — максимально допустимый размер чтения.

8. `write(1, "Sum: 3354.30\n 13) = 13`

   Результат вычислений выводится в стандартный поток вывода.

   - 1 — стандартный вывод;
   - строка с результатом;
   - 13 — длина выводимой строки.

9. `-- SIGCHLD --`

   Родительский процесс получает сигнал `SIGCHLD`, уведомляющий о завершении дочернего процесса.

10. `wait4(-1, NULL, 0, NULL) = 6474`

    Выполняется ожидание завершения дочернего процесса и освобождение его ресурсов.

    - `-1` — ожидание любого дочернего процесса;
    - `NULL` — статус завершения не сохраняется;
    - `0` — отсутствие дополнительных флагов;
    - `NULL` — информация о потреблённых ресурсах не требуется.

    Возвращён идентификатор завершившегося процесса.

Анализ системных вызовов лабораторной работы 2

```
execve("./matrix_filters", ["./matrix_filters", "1000", "1000", "5", "4"], ...)
```

Системный вызов `execve` запускает новый процесс.

- `"./matrix_filters"` — путь к исполняемому файлу.
- Массив аргументов:
  - `1000` — число строк матрицы M.
  - `1000` — число столбцов матрицы N.
  - `5` — количество повторов фильтрации K.
  - `4` — максимальное число потоков.
- `...` — указатель на переменные окружения.

```
brk(NULL)
```

Вызов `brk` запрашивает текущее положение границы кучи процесса. Используется библиотекой `glibc` для управления динамической памятью.

```
mmap(NULL, size, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0)
```

Выделение области памяти.

- `NULL` — адрес выбирается ядром автоматически.
- `size` — размер выделяемой области.
- `PROT_READ | PROT_WRITE` — разрешено чтение и запись.
- `MAP_PRIVATE` — изменения видны только процессу.

- MAP_ANONYMOUS — память не связана с файлом.
- -1, 0 — файловый дескриптор не используется.

Данный вызов используется для размещения матриц и служебных структур.

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

Загрузка стандартной библиотеки C.

- AT_FDCWD — путь задан относительно текущего каталога.
- O_RDONLY — файл открыт только для чтения.
- O_CLOEXEC — файл будет закрыт при execve.

```
mmap(...)
```

Отображение библиотеки libc.so в адресное пространство процесса.

```
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|...})
```

Создание нового потока выполнения.

- CLONE_VM — общая виртуальная память.
- CLONE_FS — общая файловая система.
- CLONE_FILES — общая таблица файловых дескрипторов.
- CLONE_SIGHAND — общие обработчики сигналов.
- CLONE_THREAD — поток в рамках одного процесса.

Этот вызов соответствует использованию pthread_create в программе.

```
futex(0x..., FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, tid, NULL, FUTEX_BITSET_MATCH_AN
```

Механизм ожидания завершения потоков. Используется библиотекой pthread для реализации pthread_join.

```
openat(AT_FDCWD, "results.csv", O_WRONLY|O_CREAT|O_TRUNC, 0666)
```

Создание файла результатов.

- O_WRONLY — запись в файл.
- O_CREAT — создать файл при отсутствии.
- O_TRUNC — очистить содержимое.
- 0666 — права доступа (rw-rw-rw-).

```
write(3, "threads,time\n1,0.372066\n...", 46)
```

Запись результатов измерений в CSV-файл.

- 3 — файловый дескриптор `results.csv`.
- Строка содержит число потоков и измеренное время.

```
write(1, "Потоков: 1, время: 0.372066 сек", 47)
```

Вывод информации в стандартный поток вывода (`stdout`).

```
close(3)
```

Закрытие файла `results.csv`.

```
exit_group(0)
```

Корректное завершение процесса и всех потоков с кодом возврата 0.

Анализ системных вызовов лабораторной работы 3

Для анализа работы программы был использован инструмент `strace`, позволяющий отследить системные вызовы, выполняемые процессом во время запуска и завершения. Ниже приведено пояснение ключевых вызовов, непосредственно связанных с выполнением программы и работой операционной системы.

`execve("./ipc_program ["./ipc_program"], envp) = 0`

Системный вызов `execve` загружает исполняемый файл программы `ipc_program` в адресное пространство текущего процесса.

- первый аргумент — путь к исполняемому файлу;
- второй аргумент — массив аргументов командной строки;
- третий аргумент — массив переменных окружения.

Возврат значения `0` означает успешный запуск программы.

`brk(NULL) = 0x5ea37c54b000`

Вызов `brk` возвращает текущее положение конца сегмента кучи (heap), который используется для динамического выделения памяти.

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

Выделяется анонимная область виртуальной памяти размером 8192 байта.

- `NULL` — адрес выбирается ядром;
- `8192` — размер отображаемой области;
- `PROT_READ | PROT_WRITE` — разрешено чтение и запись;

- MAP_PRIVATE | MAP_ANONYMOUS — приватное отображение, не связанное с файлом;
- -1 — файловый дескриптор отсутствует;
- 0 — смещение.

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

Открывается динамическая библиотека libc.so.6, необходимая для выполнения стандартных функций языка C.

mmap(..., PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED)

Библиотека отображается в виртуальную память процесса с правами на чтение и выполнение. Это стандартный этап динамической линковки.

mprotect(address, size, PROT_READ)

Системный вызов mprotect изменяет права доступа к уже отображённым страницам памяти, запрещая запись и повышая безопасность выполнения программы.

write(2, "...Использование: ./ipc_program input_file 53)

Выполняется вывод диагностического сообщения в стандартный поток ошибок.

- 2 — файловый дескриптор stderr;
- строка — сообщение об ошибке использования программы;
- 53 — количество записанных байт.

exit_group(1)

Завершение процесса и всех его потоков с кодом возврата 1, что указывает на ошибочное завершение программы.

Анализ системных вызовов лабораторной работы 4

Анализ вывода утилиты strace позволяет наглядно увидеть различия в механизмах загрузки динамических библиотек между Программой №1 (Ранняя линковка) и Программой №2 (Поздняя линковка).

Случай №1: Программа с Ранней Линковкой (./ipc_program)

Этап 1: Запуск и инициализация процесса

Процесс начинается с системных вызовов, подготавливающих среду выполнения и адресное пространство.

execve("./prog1 ["./prog1"], ...) — инициализирует выполнение программы. Ядро загружает исполняемый файл в память и передает управление динамическому загрузчику. brk(NULL) — программа запрашивает адрес конца сегмента данных. Это необходимо для последующего управления динамической памятью (кучей). mmap(NULL, 8192, PROT_READ|PROT_WRITE, ...) — выделение анонимной памяти для системных нужд (буферов ввода-вывода или служебных структур библиотек).

Этап 2: Поиск и отображение библиотек

Ключевое отличие ранней линковки заключается в том, что загрузчик (`ld-linux.so`) ищет зависимости, указанные в заголовке ELF-файла, до начала выполнения функции `main`.

- `openat(AT_FDCWD, "liblib1.so O_RDONLY|O_CLOEXEC) = 3` — загрузчик находит и открывает файл пользовательской библиотеки. `3` — файловый дескриптор открытого файла.
- `read(3, "\177ELF... 832)` — чтение заголовка ELF для проверки архитектуры и корректности библиотеки.
- `mmap(NULL, 16400, PROT_READ, ..., 3, 0)` — резервирование виртуального адресного пространства под всю библиотеку.
- `mmap(..., PROT_READ|PROT_EXEC, ..., 3, 0x1000)` — отображение секции кода библиотеки. Флаг `PROT_EXEC` разрешает процессору выполнять инструкции в этой области памяти.
- `close(3)` — закрытие файла библиотеки после завершения его отображения (`mapping`) в память.

Этап 3: Загрузка стандартной библиотеки C

После пользовательских библиотек загружаются системные зависимости, такие как `libc.so.6`.

- `openat(..., "/etc/ld.so.cache ...)` — использование системного кэша для быстрого поиска пути к стандартным библиотекам.
- `openat(..., "/lib/x86_64-linux-gnu/libc.so.6 ...)` — открытие основной библиотеки языка C, содержащей функции `printf`, `fgets` и др.

Этап 4: Настройка защиты и выполнение

Перед передачей управления пользователю, система выполняет финальные настройки безопасности.

- `mprotect(..., PROT_READ)` — перевод таблиц глобальных смещений (GOT) в режим "только чтение"для предотвращения атак на переполнение буфера.
- `arch_prctl(ARCH_SET_FS, ...)` — установка сегментного регистра для реализации локального хранилища потока (TLS).

Этап 5: Пользовательский ввод-вывод

Только после завершения всех этапов линковки мы видим системные вызовы, инициированные непосредственно кодом в `main.c`.

- `write(1, "Используется Реализация 1... 103)` — вывод приветственного сообщения в стандартный поток вывода (`stdout`, дескриптор 1).
- `read(0, ...)` — программа вызывает блокирующий системный вызов для чтения из стандартного ввода (`stdin`, дескриптор 0), ожидая команды пользователя.

Случай №2: Программа с Поздней Линковкой (`./prog2`)

Вывод `strace` для `prog2` демонстрирует ключевое отличие: использование `dlopen` для загрузки пользовательской библиотеки, что подтверждает механизм поздней линковки.

Листинг 1: Фрагмент strace для ./prog2

```
execve("./prog2", ["./prog2"], ...) = 0
openat(AT_FDCWD, "/etc/ld.so.cache", ...) = 3
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", ...) = 3
[...]
munmap(0x7ee779763000, 30255) = 0
write(1, "\320\232\320\276\320\274\320\260\320\275\320\264\321\213:\n", 16) = 16
openat(AT_FDCWD, "./liblib1.so", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0755, st_size=15208, ...}) = 0
mmap(NULL, 16400, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ee779766000
mmap(0x7ee779767000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
    3, 0x1000) = 0x7ee779767000
close(3) = 0
write(1, "\320\243\321\201\320\277\320\265\321\210\320\275\320\276
    \320\267\320\260\320\263\321\200\321\203\320\266\320\265\320\275\320"..., 58) = 58
```

Ключевые вызовы поздней линковки:

- `openat(AT_FDCWD, "./liblib1.so O_RDONLY|O_CLOEXEC) = 3`: Вызов соответствует вызову `dlopen(".liblib1.so ...)` в коде `main_dynamic.c`. Программа сама инициирует открытие файла библиотеки. Возврат 3 — это файловый дескриптор.
- `mmap(NULL, 16400, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ee779766000`: Библиотека `liblib1.so` проецируется (`mmap`) в память процесса. Аргумент 3 здесь — это файловый дескриптор, полученный от предыдущего `openat`.
- `close(3) = 0`: Файловый дескриптор, открытый для библиотеки, немедленно закрывается после того, как ее содержимое проецировано в память.
- `write(1, "204321012077206521102075 2076 206720602063..."...)`: Вывод сообщения "Успешно загружена Реализация 1. которое генерируется кодом сразу после успешного вызова `dlopen`.

Результаты

Результат вывода `strace` по лабораторной работе 1

Трассировка `strace` показывает, что программа корректно создаёт дочерний процесс, организует межпроцессное взаимодействие через неименованный канал и выполняет синхронизацию с помощью ожидания завершения дочернего процесса. Все используемые системные вызовы применяются в правильной последовательности и соответствуют логике работы лабораторной программы.

Результат вывода `strace` по лабораторной работе 2

Вывод утилиты `strace` показывает, что программа запускается как один процесс, динамически выделяет память и создаёт несколько потоков внутри этого процесса для параллельных вычислений. Многопоточность реализуется через системные вызовы `clone3` и `futex`, а результаты работы корректно записываются в файл, что подтверждает правильное использование библиотеки `pthread`.

Результат вывода `strace` по лабораторной работе 3

Результаты трассировки с помощью `strace` показывают, что программа корректно загружается в адресное пространство процесса, инициализирует виртуальную память и стандартные библиотеки, после чего переходит к выполнению пользовательского кода. Завершение работы происходит штатно с выводом диагностического сообщения, что подтверждает корректную обработку ошибок и взаимодействие программы с операционной системой.

Результат вывода `strace` по лабораторной работе 4

Анализ системных вызовов `strace` подтвердил фундаментальное различие в механизмах загрузки: в то время как системные библиотеки (`libc.so.6`) загружаются системным загрузчиком автоматически, `prog2` демонстрирует **позднюю линковку**, программно инициируя вызовы `openat` и `mmap` для загрузки `liblib1.so` уже в процессе выполнения.

Выводы

В ходе выполнения цикла лабораторных работ были успешно освоены фундаментальные механизмы системного программирования и взаимодействия программ с операционной системой. Анализ системных вызовов `strace` подтвердил освоение различных форм параллельной обработки: межпроцессное взаимодействие (ЛР 1) было реализовано через системные вызовы `fork`, `pipe` и `wait` для создания процессов и синхронизации через каналы; в то время как многопоточность (ЛР 2) использовала низкоуровневые вызовы `clone3` и `futex` для управления потоками внутри единого адресного пространства. Одновременно были изучены методы управления кодом и его загрузкой (ЛР 4). Был продемонстрирован контраст между автоматической загрузкой системных библиотек (`libc.so.6`) системным загрузчиком и программным управлением поздней линковкой, где тестовая программа `prog2` инициировала вызовы `openat` и `mmap` для загрузки `liblib1.so` уже во время выполнения. Это подтверждает успешное освоение интерфейса `dlfcn` для реализации модульной архитектуры и плагинов. Наконец, базовая трассировка (ЛР 3) показала корректный процесс инициализации программы, включая выделение виртуальной памяти и штатную обработку системных зависимостей. Таким образом, был освоен полный цикл взаимодействия программы с ОС: от базовой загрузки и выделения ресурсов до реализации сложных механизмов параллелизма и динамического управления кодом.

Исходная программа

Strace Лабораторной работы 1

```
execve("./parent", ["./parent"], 0x7ffe01b3e6a0 /* 27 vars */) = 0
```

```
brk(NULL)                               = 0x64ec69cf7000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x742b4faa
access("/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=30327, ...}) = 0
mmap(NULL, 30327, PROT_READ, MAP_PRIVATE, 3, 0) = 0x742b4faa3000
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\0\220\243\2\0\0\0\0\0"..., 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, 6
fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 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, 6
mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x742b4f800000
mmap(0x742b4f828000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRIT
mmap(0x742b4f9b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0
mmap(0x742b4f9ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE
mmap(0x742b4fa05000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x742b4fa
arch_prctl(ARCH_SET_FS, 0x742b4faa0740) = 0
set_tid_address(0x742b4faa0a10)         = 19647
set_robust_list(0x742b4faa0a20, 24)     = 0
rseq(0x742b4faa1060, 0x20, 0, 0x53053053) = 0
mprotect(0x742b4f9ff000, 16384, PROT_READ) = 0
mprotect(0x64ec2c20e000, 4096, PROT_READ) = 0
mprotect(0x742b4fae3000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x742b4faa3000, 30327)           = 0
pipe2([3, 4], 0)                        = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
getrandom("\x85\x78\x9f\xdd\x02\xb7\x3f\x3c", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x64ec69cf7000
brk(0x64ec69d18000)                     = 0x64ec69d18000
write(1, "Filename:\n", 10Filename:
)             = 10
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
read(0, input.txt
"input.txt\n", 1024)            = 10
openat(AT_FDCWD, "input.txt", O_RDONLY) = 5
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_
close(4)                                = 0
close(5)                                = 0
read(3, "", 4096)                       = 0
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=19666, si_uid=1000, si_stat
close(3)                                = 0
wait4(-1, NULL, 0, NULL)                = 19666
lseek(0, -1, SEEK_CUR)                  = -1 ESPIPE (Illegal seek)
exit_group(0)                           = ?
```

Strace Лабораторной работы 2

```
execve("./matrix_filters", ["./matrix_filters"], 0x7fffcb8c7610 /* 27 vars */) = 0
brk(NULL)                                = 0x62e18077a000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ad1baf6
access("/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=30327, ...}) = 0
mmap(NULL, 30327, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ad1baf5b000
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\0\220\243\2\0\0\0\0\0"..., 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, 6
fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 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, 6
mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ad1bac00000
mmap(0x7ad1bac28000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRIT
mmap(0x7ad1badb0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0
mmap(0x7ad1badff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE
mmap(0x7ad1bae05000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS
close(3)                                 = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ad1baf
arch_prctl(ARCH_SET_FS, 0x7ad1baf58740) = 0
set_tid_address(0x7ad1baf58a10)          = 20086
set_robust_list(0x7ad1baf58a20, 24)      = 0
rseq(0x7ad1baf59060, 0x20, 0, 0x53053053) = 0
mprotect(0x7ad1badff000, 16384, PROT_READ) = 0
mprotect(0x62e151263000, 4096, PROT_READ) = 0
mprotect(0x7ad1baf9b000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7ad1baf5b000, 30327)            = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
getrandom("\x16\x48\x0f\x99\x97\x9d\x01\xe4", 8, GRND_NONBLOCK) = 8
brk(NULL)                                = 0x62e18077a000
brk(0x62e18079b000)                      = 0x62e18079b000
write(1, "\320\230\321\201\320\277\320\276\320\273\321\214\320\267\320\276\320\262\32
) = 56
exit_group(1)                            = ?
```

Strace Лабораторной работы 3

```
execve("./ipc_program", ["./ipc_program"], 0x7ffc644f3670 /* 27 vars */) = 0
brk(NULL)                                = 0x5fef289e2000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7a2e6e1b
access("/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=30327, ...}) = 0
mmap(NULL, 30327, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7a2e6e1ae000
```

```
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\0\220\243\2\0\0\0\0\0"..., 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, 6
fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 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, 6
mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7a2e6de00000
mmap(0x7a2e6de28000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRIT
mmap(0x7a2e6dfb0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0
mmap(0x7a2e6dfff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE
mmap(0x7a2e6e005000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7a2e6e1
arch_prctl(ARCH_SET_FS, 0x7a2e6e1ab740) = 0
set_tid_address(0x7a2e6e1aba10)         = 20363
set_robust_list(0x7a2e6e1aba20, 24)     = 0
rseq(0x7a2e6e1ac060, 0x20, 0, 0x53053053) = 0
mprotect(0x7a2e6dfff000, 16384, PROT_READ) = 0
mprotect(0x5fef1a6c0000, 4096, PROT_READ) = 0
mprotect(0x7a2e6e1ee000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7a2e6e1ae000, 30327)           = 0
write(2, "\320\230\321\201\320\277\320\276\320\273\321\214\320\267\320\276\320\262\32
) = 53
exit_group(1)                           = ?
```

Strace Лабораторной работы 4

```
\subsection*{Для prog1}
\begin{verbatim}
execve("./ipc_program", ["./ipc_program"], 0x7ffcca0c1630 /* 27 vars */) = 0
brk(NULL)                               = 0x5ea37c54b000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x76058ef9
access("/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=30255, ...}) = 0
mmap(NULL, 30255, PROT_READ, MAP_PRIVATE, 3, 0) = 0x76058ef97000
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\0\220\243\2\0\0\0\0\0"..., 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, 6
fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 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, 6
mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x76058ec00000
mmap(0x76058ec28000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRIT
mmap(0x76058edb0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0
mmap(0x76058edff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE
mmap(0x76058ee05000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS
```

```
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x76058ef
arch_prctl(ARCH_SET_FS, 0x76058ef94740) = 0
set_tid_address(0x76058ef94a10)         = 1482
set_robust_list(0x76058ef94a20, 24)     = 0
rseq(0x76058ef95060, 0x20, 0, 0x53053053) = 0
mprotect(0x76058edff000, 16384, PROT_READ) = 0
mprotect(0x5ea33ed8e000, 4096, PROT_READ) = 0
mprotect(0x76058efd7000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x76058ef97000, 30255)           = 0
write(2, "\320\230\321\201\320\277\320\276\320\273\321\214\320\267\320\276\320\262\32
) = 53
exit_group(1)                           = ?
```

Для prog2

```
execve("./prog2", ["./prog2"], 0x7ffc688ed910 /* 27 vars */) = 0
brk(NULL)                               = 0x62750638a000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ee77976
access("/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=30255, ...}) = 0
mmap(NULL, 30255, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ee779763000
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\0\220\243\2\0\0\0\0\0"..., 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, 6
fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 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, 6
mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ee779400000
mmap(0x7ee779428000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRIT
mmap(0x7ee7795b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0
mmap(0x7ee7795ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE
mmap(0x7ee779605000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ee7797
arch_prctl(ARCH_SET_FS, 0x7ee779760740) = 0
set_tid_address(0x7ee779760a10)         = 9490
set_robust_list(0x7ee779760a20, 24)     = 0
rseq(0x7ee779761060, 0x20, 0, 0x53053053) = 0
mprotect(0x7ee7795ff000, 16384, PROT_READ) = 0
mprotect(0x6274fb7c5000, 4096, PROT_READ) = 0
mprotect(0x7ee7797a3000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7ee779763000, 30255)           = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
```

```
getrandom("\x55\xd2\x7f\xbd\x4f\x65\xd2\xcb", 8, GRND_NONBLOCK) = 8
brk(NULL)                                       = 0x62750638a000
brk(0x6275063ab000)                             = 0x6275063ab000
write(1, "\320\232\320\276\320\274\320\260\320\275\320\264\321\213:\n", 16Команды:
) = 16
write(1, "0 : \320\237\320\265\321\200\320\265\320\272\320\273\321\216\321\207\320\26
) = 61
write(1, "1 arg1...: \320\222\321\213\320\267\320\276\320\262 Pi(K).\n", 291 arg1...:
) = 29
write(1, "2 arg1 arg2...: \320\222\321\213\320\267\320\276\320\262 Squar"..., 412 arg
) = 41
openat(AT_FDCWD, "./liblib1.so", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 8
fstat(3, {st_mode=S_IFREG|0755, st_size=15208, ...}) = 0
getcwd("/home/b0rus1a/OSLabs/Lab4", 128) = 26
mmap(NULL, 16400, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ee779766000
mmap(0x7ee779767000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
mmap(0x7ee779768000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000)
mmap(0x7ee779769000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
close(3)                                        = 0
mprotect(0x7ee779769000, 4096, PROT_READ) = 0
write(1, "\320\243\321\201\320\277\320\265\321\210\320\275\320\276 \320\267\320\260\3
) = 58
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
write(1, ">>> ", 4>>> )                      = 4
```