

**НЕГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ ЧАСТНОЕ** **УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ** **«МОСКОВСКИЙ ФИНАНСОВО-ПРОМЫШЛЕННЫЙ УНИВЕРСИТЕТ** **«СИНЕРГИЯ»**

|  |  |  |
| --- | --- | --- |
| **Факультет/Институт** |  | информационных технологий |
|  |  | (наименование факультета/ Института) |
| **Направление/специальность** |  | 09.02.07 Информационные системы и программирование |
| **подготовки:** |  | (код и наименование направления /специальности подготовки) |
| **Форма обучения:** |  | очная |
|  |  | (очная, очно-заочная, заочная) |
|  |  |  |

.

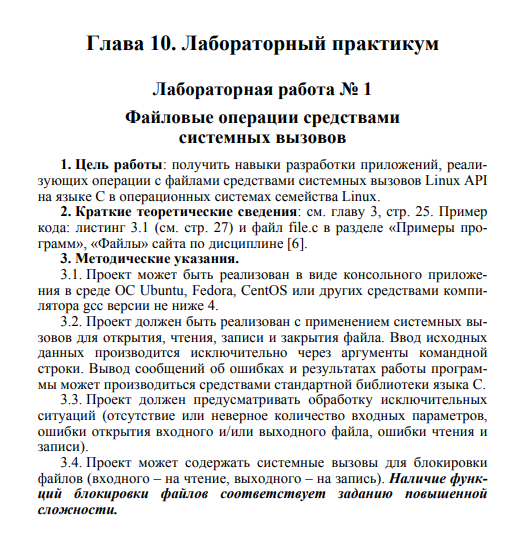
**Отчет по лабораторной работе № 2**

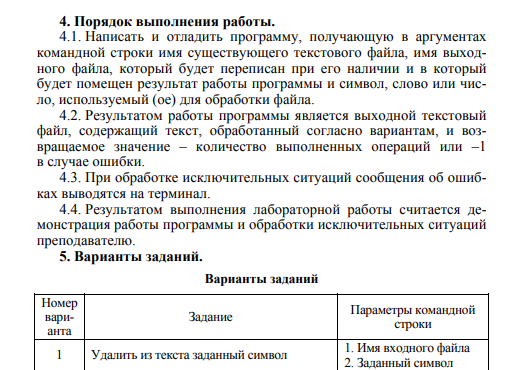
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **на тему** | |  | Переменные окружения | | | | |
|  | | | | | |  | (наименование темы) |
|  |  | |  | | | | |
| **по дисциплине** | | | |  | Системное программирование | | |
|  | | | | | |  | (наименование дисциплины) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Обучающийся** |  | Ценев Алексей Константинович |  |  |
|  |  | (ФИО) |  | (подпись) |
| **Группа** |  | ДКИП-206 Прог |  |
|  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Преподаватель** |  |  |  |  |
|  |  | (ФИО) |  | (подпись) |

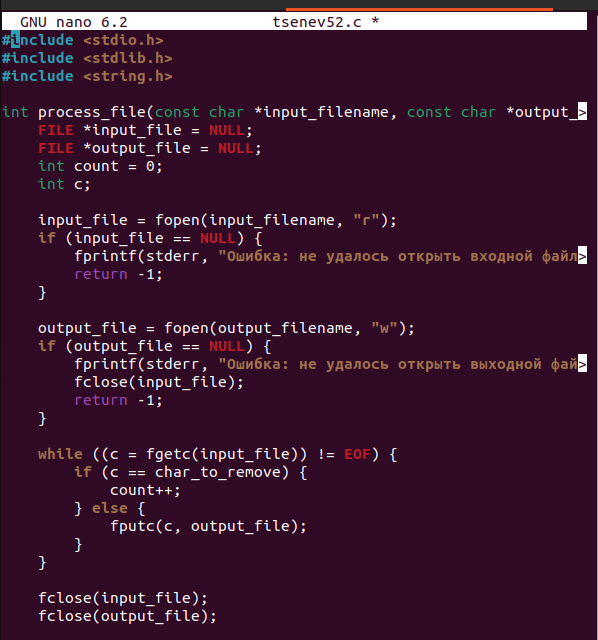
**Москва, 2025**

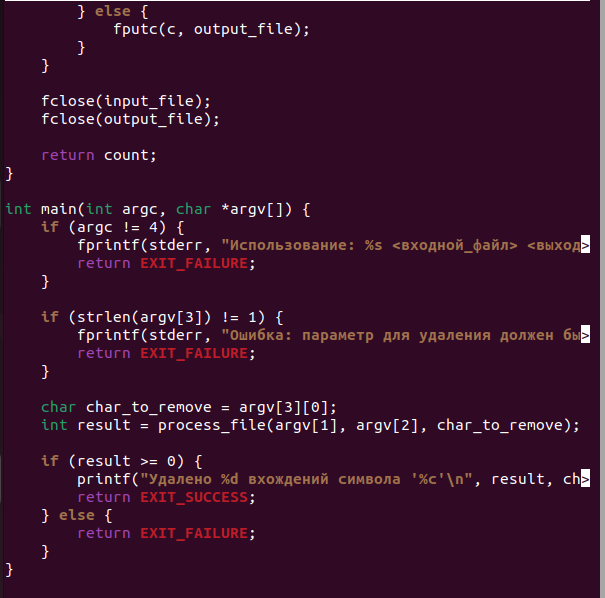


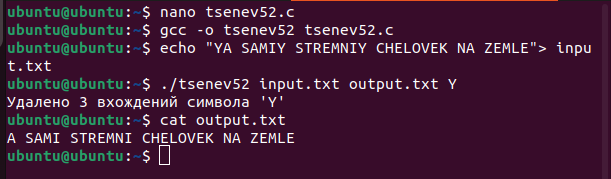


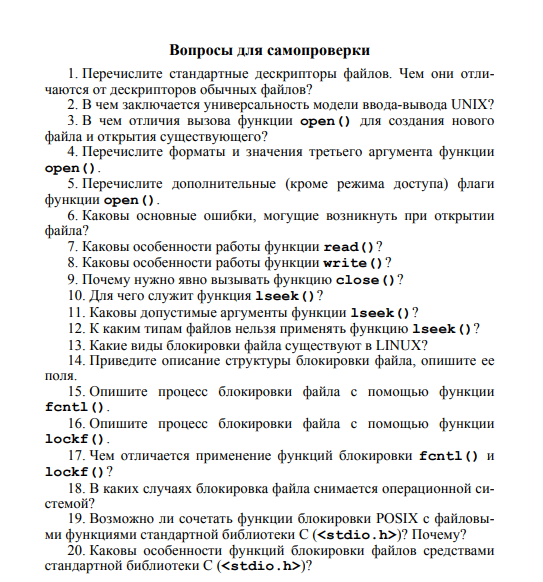
Вариант 1











**1. Стандартные дескрипторы файлов**

Стандартные дескрипторы файлов:

* 0 (STDIN\_FILENO) - стандартный ввод
* 1 (STDOUT\_FILENO) - стандартный вывод
* 2 (STDERR\_FILENO) - стандартный вывод ошибок

От обычных дескрипторов они отличаются тем, что автоматически открываются при запуске процесса и связаны с терминалом по умолчанию.

**2. Универсальность модели ввода-вывода UNIX**

Универсальность заключается в том, что все устройства и файлы представлены как последовательности байтов, и для работы с ними используются одни и те же системные вызовы (open, read, write, close).

**3. Отличия open() для нового и существующего файла**

При создании нового файла:

* Нужно указать флаг O\_CREAT
* Требуется указать права доступа (mode)
* Если файл уже существует, может понадобиться O\_EXCL для предотвращения перезаписи

При открытии существующего файла флаг O\_CREAT не обязателен, а параметр mode игнорируется.

**4. Форматы и значения третьего аргумента open()**

Третий аргумент (mode) используется только с O\_CREAT и определяет права доступа:

* S\_IRWXU - права пользователя (чтение/запись/исполнение)
* S\_IRUSR, S\_IWUSR, S\_IXUSR - отдельные права пользователя
* S\_IRWXG - права группы
* S\_IRGRP, S\_IWGRP, S\_IXGRP - отдельные права группы
* S\_IRWXO - права других
* S\_IROTH, S\_IWOTH, S\_IXOTH - отдельные права других

**5. Дополнительные флаги open()**

* O\_APPEND - запись в конец файла
* O\_ASYNC - асинхронный ввод-вывод
* O\_CLOEXEC - закрыть при exec
* O\_DIRECT - минимизировать кэширование
* O\_DIRECTORY - ошибка если не директория
* O\_DSYNC - синхронизация данных
* O\_EXCL - эксклюзивное создание
* O\_NOATIME - не обновлять время доступа
* O\_NOCTTY - не делать терминал управляющим
* O\_NOFOLLOW - не следовать по симлинкам
* O\_NONBLOCK - неблокирующий режим
* O\_RSYNC - синхронизация чтения
* O\_SYNC - синхронизация файла
* O\_TMPFILE - создать безымянный временный файл
* O\_TRUNC - обрезать файл до нулевой длины

**6. Основные ошибки при открытии файла**

* EACCES - нет прав доступа
* EEXIST - файл существует (с O\_CREAT|O\_EXCL)
* EISDIR - файл является директорией
* EMFILE - достигнут лимит открытых файлов
* ENFILE - системный лимит файлов
* ENOENT - файл не существует (без O\_CREAT)
* ENOSPC - нет места на устройстве
* EROFS - попытка записи в RO-ФС

**7. Особенности read()**

* Возвращает количество прочитанных байт
* Для обычных файлов читает запрошенное количество байт
* Для терминалов и сетевых сокетов может читать меньше
* Возвращает 0 при достижении конца файла
* Может быть прерван сигналом (EINTR)

**8. Особенности write()**

* Возвращает количество записанных байт
* Может записать меньше запрошенного (например, при нехватке места)
* Для сокетов и каналов может потребовать нескольких вызовов
* Может быть прерван сигналом (EINTR)

**9. Причины явного вызова close()**

* Освобождение дескриптора (ограниченный ресурс)
* Сброс буферов на диск
* Разблокировка файла для других процессов
* Освобождение ресурсов ядра

**10. Назначение lseek()**

Изменяет позицию чтения/записи в файле:

* Позволяет произвольный доступ к файлу
* Используется для определения текущей позиции
* Может расширять файл (с O\_APPEND)

**11. Допустимые аргументы lseek()**

* offset - смещение
* whence - точка отсчета:
  + SEEK\_SET - от начала файла
  + SEEK\_CUR - от текущей позиции
  + SEEK\_END - от конца файла

**12. Типы файлов, для которых нельзя использовать lseek()**

* Каналы (pipes)
* FIFO (именованные каналы)
* Сокеты
* Терминалы и другие символьные устройства

**13. Виды блокировки файлов в Linux**

* Обязательные (mandatory) - ядро обеспечивает блокировку
* Консультативные (advisory) - только соглашение между процессами
* Чтение (shared) - несколько читателей
* Запись (exclusive) - только один писатель

**14. Структура блокировки файла**

struct flock {  
short l\_type; // F\_RDLCK, F\_WRLCK, F\_UNLCK  
short l\_whence; // SEEK\_SET, SEEK\_CUR, SEEK\_END  
off\_t l\_start; // смещение  
off\_t l\_len; // длина (0 = до EOF)  
pid\_t l\_pid; // PID блокирующего процесса  
};

**15. Блокировка через fcntl()**

1. Создать структуру flock
2. Установить параметры блокировки
3. Вызвать fcntl() с командой:
   * F\_SETLK - неблокирующая попытка
   * F\_SETLKW - блокирующая
   * F\_GETLK - проверка возможности

**16. Блокировка через lockf()**

Упрощенный интерфейс поверх fcntl():

* F\_LOCK - блокирующая эксклюзивная блокировка
* F\_TLOCK - неблокирующая эксклюзивная
* F\_ULOCK - разблокировка
* F\_TEST - проверка возможности блокировки

**17. Отличия fcntl() и lockf()**

* lockf() проще, но менее гибкая
* lockf() всегда работает с текущей позицией до EOF
* fcntl() позволяет точнее задавать диапазон
* fcntl() поддерживает разделяемые блокировки

**18. Снятие блокировки ОС**

* При закрытии файла (в большинстве случаев)
* При завершении процесса
* В некоторых случаях при exec()
* При перезагрузке системы

**19. Сочетание POSIX и stdio**

Не рекомендуется, так как:

* Стандартные библиотечные функции буферизируют данные
* Позиция в файле может расходиться
* Блокировки могут не учитываться

**20. Особенности блокировок в stdio**

* Функции типа flockfile()/funlockfile() работают только в пределах одного процесса
* Обеспечивают потокобезопасность
* Не влияют на другие процессы
* Работают с FILE\*, а не с файловыми дескрипторами