**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**
**Федеральное государственное автономное образовательное учреждение высшего образования**
**«Новосибирский национальный исследовательский государственный университет»**
**(Новосибирский государственный университет, НГУ)**

**Физический факультет**
**Кафедра физико-технической информатики**

УТВЕРЖДАЮ
Декан ФФ, д.ф.-м.н
_____ В.Е.Блинов
«08» 08 2022 г.

**Рабочая программа дисциплины**

# ОПЕРАЦИОННЫЕ СИСТЕМЫ

**направление подготовки: 03.03.02 Физика**
**Направленность (профиль): Физическая информатика**

Форма обучения
**Очная**

| **Семестр** | **Общий объем** | **Виды учебных занятий (в часах)** | | | | **Промежуточная аттестация (в часах)** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Контактная работа обучающихся с преподавателем | | | Самостоятельная работа, не включая период сессии | Самостоятельная подготовка к промежуточной аттестации | Контактная работа обучающихся с преподавателем | | | |
| | | Лекции | Практические занятия | Консультации в период занятий | | | Консультации | Зачет | Дифференцированный зачет | Экзамен |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 5 | 108 | 32 | 32 | | 20 | 18 | 4 | | | 2 |
| Всего 108 часов / 3 зачётные единицы, из них:<br>- контактная работа 70 часов | | | | | | | | | | |
| Компетенции ОПК-3 | | | | | | | | | | |

Ответственный за образовательную программу
д.ф.-м.н., проф. С. В. Цыбуля

**Новосибирск, 2022**

**Содержание**

## 1. Перечень планируемых результатов обучения по дисциплине, соотнесённых с планируемыми результатами освоения образовательной программы.

Дисциплина (курс) «Операционные системы» имеет своей целью ознакомление с основами и принципами построения и функционирования современных электронно-вычислительных машин (компьютеров), определяющих их применение для тех или иных целей.

Дисциплина нацелена на формирование у обучающихся общепрофессиональной компетенции ОПК-3.

| Результаты освоения образовательной программы (компетенции) | Индикаторы | Результаты обучения по дисциплине |
|---|---|---|
| **ОПК-3.** Способен понимать принципы работы современных информационных технологий и использовать их для решения задач профессиональной деятельности | **ОПК - 3.1.** Применяет различные источники информации для решения задач профессиональной сферы деятельности.<br><br>**ОПК – 3.2.** Применяет основные приемы, возможности и правила работы со стандартными и специализированными программными продуктами при решении профессиональных задач.<br><br>**ОПК – 3.3.** Применяет методологию поиска научной и технической информации в сети Интернет и специализированных базах данных.<br>. | **Знать** теоретические основы, используемые при построении современных операционных систем; современное состояние компьютерных технологий, используемых для научных и практических целей.<br><br>**Уметь** выбирать программное обеспечение для решения конкретной проблемы; работать с программными интерфейсами системных вызовов и библиотек, соответствующих стандарту POSIX. |

## 2. Место дисциплины в структуре образовательной программы.

Дисциплина «Операционные системы» реализуется в осеннем семестре 3-го курса бакалавриата для обучающихся по направлению подготовки **03.03.02 Физика**. Курс реализуется кафедрой физико-технической информатики.

Для достижения поставленной цели выделяются задачи курса:

1. изучение теоретических основ построения операционных систем;
2. сравнительное рассмотрение архитектур ОС разных семейств;
3. обоснованный выбор и использование современных компьютерных архитектур.

**3. Трудоёмкость дисциплины в зачётных единицах с указанием количества академических часов, выделенных на контактную работу обучающегося с преподавателем (по видам учебных занятий) и на самостоятельную работу.**

| Семестр | Общий объем | Виды учебных занятий (в часах) | | | | Промежуточная аттестация (в часах) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Контактная работа обучающихся с преподавателем | | | Самостоятельная работа, не включая период сессии | Самостоятельная подготовка к промежуточной аттестации | Контактная работа обучающихся с преподавателем | | | |
| | | Лекции | Практические занятия | Консультации в период занятий | | | Консультации | Зачет | Дифференцированный зачет | Экзамен |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 5 | 108 | 32 | 32 | | 20 | 18 | 4 | | | 2 |
| Всего 108 часов / 3 зачётные единицы, из них:<br>- контактная работа 70 часов | | | | | | | | | | |
| Компетенции ОПК-3 | | | | | | | | | | |

Реализация дисциплины предусматривает практическую подготовку при проведении следующих видов занятий, предусматривающих участие обучающихся в выполнении отдельных элементов работ, связанных с будущей профессиональной деятельностью: лекции, лабораторные занятия, самостоятельная работа студента и её контроль преподавателями с помощью заданий, консультации, экзамен.

Программой дисциплины предусмотрены следующие виды контроля:

- текущий контроль успеваемости: задания на лабораторных работах;
- промежуточная аттестация: экзамен.

Общая трудоёмкость рабочей программы дисциплины составляет 3 зачётные единицы.

- занятия лекционного типа – 32 часа;
- практические занятия – 32 часа;
- самостоятельная работа обучающегося в течение семестра, не включая период сессии – 20 часов;
- промежуточная аттестация (подготовка к экзамену, консультации и экзамен) – 24 часа.

Объём контактной работы обучающегося с преподавателем (занятия лекционного типа, практические занятия, консультации, экзамен) составляет 70 часов.

**4. Содержание дисциплины, структурированное по темам (разделам) с указанием отведённого на них количества академических часов и видов учебных занятий.**

Общая трудоёмкость дисциплины составляет 3 зачётные единицы, 108 академических часов.

| № п/п | Раздел дисциплины | Неделя семестра | Виды учебной работы, включая самостоятельную работу студентов и трудоёмкость (в часах) | | | | | | Консультации перед экзаменом (часов) | Промежуточная аттестация (в часах) |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Всего | Аудиторные часы | | | Сам. работа во время занятий (не включая период сессии) | Сам. работа во время промежуточной аттестации | | |
| | | | | Лекции | Практические занятия | Консультации в период занятий | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 1. | Среда исполнения Unix | 1 | 7 | 2 | 4 | 1 | | | | |
| 2. | Организация памяти. Управление памятью на уровне процесса. Виртуальная память. | 2-3 | 8 | 4 | 2 | | 2 | | | |
| 3. | Файловый ввод-вывод. Отображение файлов на память | 4 | 9 | 2 | 6 | | 1 | | | . |
| 4. | Создание процессов и исполнение программ | 5 | 9 | 4 | 4 | | 1 | | | |
| 5. | Загрузка операционной системы | 6 | 3 | 2 | | | 1 | | | |
| 6. | Управление файлами и каталогами | 7-8 | 7 | 2 | 4 | | 1 | | | |
| 7. | Терминальный ввод-вывод | 9 | 5 | 2 | 2 | | 1 | | | |
| 8. | Сигналы | 10 | 5 | 2 | 2 | | 1 | | | |
| 9. | Средства межпроцессного взаимодействия: трубы, сокеты Unix и TCP/UDP | 11-12 | 12 | 4 | 6 | | 2 | | | |
| 10. | Взаимодействие через разделяемую память. Примитивы синхронизации<br>Гармоническое взаимодействие<br>Событийно-ориентированные архитектуры | 13 | 4 | 2 | | | 2 | | | |
| 11. | Файловые системы. | 14 | 4 | 2 | | | 2 | | | |
| 12. | Вопросы безопасности | 15 | 6 | 2 | 2 | | 2 | | | |
| 13. | Контейнерная и гипервизорная виртуализация | 16 | 4 | 2 | | | 2 | | | |
| 14. | Промежуточная аттестация (экзамен) | | 24 | | | | | 18 | 4 | 2 |
| **Всего** | | | **108** | **32** | **32** | | **20** | **18** | **4** | **2** |

**Программа и основное содержание лекций (32 часа)**

1. Среда исполнения. Ядро и пользовательское окружение (userland). Различие между функцией и системным вызовом. Статическая и динамическая сборка. Структура пользовательского адресного пространства в Solaris x86 и x64. Атрибуты процесса (uid/gid, euid/egid, pid, pgid, sid, get/setrlimit). Переменные среды. **(2 часа)**
2. Организация памяти. Управление памятью. Malloc(3C)/free(3C), alloca и динамические массивы C99. Стратегии управления памятью: first, best, worst fit. Внешняя и внутренняя фрагментация. Стек и стековый кадр. Алгоритм парных меток. Обзор реализаций malloc в реальных библиотеках языка C. Алгоритм близнецов. Слабовый аллокатор. Сборка мусора подсчетом ссылок. Сборка мусора mark'n'sweep. Копирующие и генерационные сборщики мусора. Обзор проблем при взаимодействии сред исполнения с разными стратегиями сборки мусора. Страничная и сегментная виртуальная память. Страничная подкачка. Использование подкачки для реализации mmap. Алгоритмы поиска жертвы при страничной подкачке. Копирование при записи и его использование для реализации fork. **(4 часа)**
3. Файловый ввод-вывод. Системные вызовы ввода/вывода. Файлы. Понятие файла в Unix. Системный вызов open(2). Системные вызовы read/readv, write/writev. Системный вызов lseek. Разреженные файлы. Вызовы fsync, dup, fcntl. Мультиплексирование ввода/вывода (select/poll). Отображение файлов на память. **(2 часа)**
4. Создание процессов и исполнение программ. Системные вызовы fork и exec. Статус завершения процесса. Понятие зомби. Системные вызовы wait, waitid, waitpid. Дополнительно: функции dlopen и dlsym. Кооперативные и вытесняющие планировщики. Реальное и разделенное время. Традиционные планировщики разделенного времени (динамические приоритеты). Справедливые планировщики. **(4 часа)**
5. Загрузка операционной системы. Загрузочное ПЗУ (bootloader). Первичный загрузчик. Вторичный загрузчик. Загрузка модулей ядра. BIOS PC-совместимых компьютеров. UEFI. Secure boot. Другие схемы защиты на уровне загрузчика. **(2 часа)**
6. Управление файлами и каталогами. Атрибуты файла. Системный вызов stat. Традиционные права доступа Unix. Setuid. БД учетных записей. Функции getpwent/getpwnam/getpwuid. Чтение содержимого каталога. Библиотечные функции opendir/readdir. Жесткие и символические ссылки. Удаление и переименование файла. **(2 часа)**
7. Терминальный ввод-вывод. Терминальные устройства в Unix. Физические терминалы. Псевдотерминалы. Стандартизованный интерфейс для управления терминальным интерфейсом tcgetattr/tscetattr. Канонический и неканонический режимы. Управление заданиями. **(2 часа)**
8. Сигналы. Традиционные (ненадежные) сигналы в ОС семейства Unix. Обработка сигнала. Генерация сигналов. Будильники, системный вызов kill. Функции setjmp/longjmp. Маска сигналов. **(2 часа)**
9. Средства межпроцессного взаимодействия: трубы и сокеты. Неименованные трубы. Именованные трубы. Unix domain sockets. Сокеты TCP. Дополнительно: сокеты UDP. **(4 часа)**
10. Взаимодействие через разделяемую память. Критические секции и синхронизация. Определение критической секции. Флаговая переменная как простейшая критическая секция. Алгоритм Деккера. CAS и спинлок. Атомарные примитивы (атомарный инкремент и др.). Полная и частичная сериализация и ее влияние на производительность. Семафоры и семафороподобные примитивы. Реентерабельные функции и thread-safe API. Мертвые и живые блокировки. Инверсия приоритета.
    Гармонически взаимодействующие процессы. Понятие гармонически взаимодействующих процессов. Примитивы гармонического взаимодействия: буферизованные (трубы и очереди сообщений) и небуферизованные (линки и рандеву языка Ada). Реализация произвольного доступа к данным в рамках гармонического взаимодействия. Языки запросов. Использование буферизованных примитивов для обхода инверсии приоритета. Событийно-ориентированные архитектуры. Голливудский принцип ("не звоните нам, вы вам

позвоним"). Применения событийно-ориентированных архитектур: графические интерфейсы, подсистема ввода-вывода, сетевые серверы. Преимущества событийно-ориентированной архитектуры перед пулами потоков. Недостатки событийно-ориентированной архитектуры. **(2 часа)**

11. Файловые системы. Драйверы устройств. Понятие драйвера и псевдоустройства. Интерфейс драйвера в традиционных ОС семейства Unix. Блочные и символьные устройства. Идеология "Всё - файл". Ioctl. Терминальный интерфейс Unix как пример провала идеологии "всё - файл" и терминальные ioctl как пример попытки обхода этого провала. Альтернативные подходы к построению интерфейса драйверов. Обзор Windows Driver Model. Проблема отображения имен на адреса. Понятие каталога. Простые ФС: tar, RT-11, ISO9660, FAT. Файловые системы с инодами (метафайлом). Обзор структур ufs/ext3, NTFS, ext4. Восстановление целостности после сбоя. Журнальные файловые системы. Файловые системы с копированием при записи: NetApp WAFL, ZFS. Сетевые и псевдофайловые системы на примере /proc. **(2 часа)**
12. Вопросы безопасности. Модели управления правами доступа: ACL, роли, полномочия. Принцип минимума привилегий. Традиционные права доступа Unix как ACL фиксированной структуры. Кольца доступа. ACL произвольного вида. UID как полномочие (capability). Аутентификация по токену в Win32. Парольная аутентификация. Дополнительно: Аутентификация запрос-ответ (CHAP). Протокол ssh. Сетевые БД учетных записей. Схема Нидхама-Шредера. Kerberos. Использование шифрования с открытым ключом. PKI. **(2 часа)**
13. Контейнерная и гипервизорная виртуализация. Контейнерная виртуализация: зоны Solaris, VPS OpenVZ/Virtuozzo. Docker. Теорема Попека/Голдберга. VM/370. Подходы к реализации VM на машинах, не соответствующих теореме Попека/ Голдберга. Паравиртуализация. Покомандная эмуляция. Jit-компиляция. **(2 часа)**

**Программа практических занятий (32 часа)**

1. Введение в Unix. **(2 часа)**
2. Среда исполнения. Ядро и пользовательское окружение (userland). Различие между функцией и системным вызовом. **(2 часа)**
3. Атрибуты процесса (uid/gid, euid/egid, pid, pgid, sid, get/setrlimit). Переменные . **(2 часа)**
4. Управление памятью. Malloc(3C)/free(3C), alloca и динамические массивы C99. **(2 часа)**
5. Системные вызовы ввода/вывода. Файлы. Понятие файла в Unix. Системный вызов open(2). Системные вызовы read/readv, write/writev. Системный вызов lseek. Вызовы fsync, dup, fcntl. Мультиплексирование ввода/вывода (select/poll). Отображение файлов на память. **(4 часа)**
6. Создание процессов и исполнение программ. Системные вызовы fork и exec. Статус завершения процесса. Понятие зомби. Системные вызовы wait, waitid, waitpid. **(4 часа)**
7. Управление файлами и каталогами. Атрибуты файла. Системный вызов stat. Традиционные права доступа Unix. Setuid. **(4 часа)**
8. БД учетных записей. Функции getpwent/getpwnam/getpwuid. **(2 часа)**
9. Чтение содержимого каталога. Библиотечные функции opendir/readdir. Жесткие и символические ссылки. Удаление и переименование файла. **(2 часа)**
10. Терминальный ввод-вывод. Терминальные устройства в Unix. Физические терминалы. Псевдотерминалы. Стандартизованный интерфейс для управления терминальным интерфейсом tcgetattr/tscetattr. Канонический и неканонический режимы. **(2 часа)**
11. Управление заданиями. **(2 часа)**
12. Сигналы. Традиционные (ненадежные) сигналы в ОС семейства Unix. Обработка сигнала. Генерация сигналов. Будильники, системный вызов kill. Функции setjmp/longjmp. Маска сигналов. **(2 часа)**
13. Трубы и сокеты. Неименованные трубы. Именованные трубы. Unix domain sockets. Сокеты TCP. Дополнительно: сокеты UDP. **(2 часа)**

**Самостоятельная работа студентов (38 часов)**

| Перечень занятий на СРС | Объем, час |
|---|---|
| Подготовка к лабораторным занятиям | 20 |
| Подготовка к экзамену | 18 |

## 5. Перечень учебной литературы.


1. Д. В. Иртегов. Введение в операционные системы: [учебное пособие для вузов по направлению 230100 "Информатика и вычислительная техника"] / 2-е изд., [перераб. и доп.]. - Санкт-Петербург: БХВ-Петербург, 2008. - 1040 с. (80 экз.)
2. Э. Таненбаум. Современные операционные системы = Modern Operating Systems: [пер. с англ.] / 2-е изд. - СПб. и др.: ПИТЕР, 2007. - 1037 с. (59 экз.)


## 6. Перечень учебно-методических материалов по самостоятельной работе обучающихся.


3. Д. В. Иртегов. Многопоточное программирование с использованием POSIX Thread Librari: учебное пособие: [для студентов ФИТ НГУ] / Федер. агентство по образованию, Новосиб. гос. ун-т, Фак. информ. технологий. Новосибирск: Редакционно-издательский центр НГУ, 2008. - 158 с. (44 экз.)


## 7. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины.

### 7.1 Ресурсы сети Интернет

Для освоения дисциплины используются следующие ресурсы:
- электронная информационно-образовательная среда НГУ (ЭИОС);
- образовательные интернет-порталы;
- информационно-телекоммуникационная сеть Интернет.

Интернет-ресурсы:
Список задач для практикума: http://ccfit.nsu.ru/~fat/svr4tasks-new.html

| № п/п | Наименование Интернет-ресурса | Краткое описание |
|---|---|---|
| 1. | http://openindiana.org/ | Open Solaris 11 (бесплатное программное обеспечение) |
| 2. | https://www.oracle.com/search/results?cat=otn&Ntk=S3&Ntt=manual | Встроенное системное руководство man (также входит в состав системы) |
| 3. | http://www.oracle.com/technetwork/server-storage/solarisstudio/downloads/index-jsp-138519.html | Oracle Solaris Studio (доступен бесплатно после регистрации на сайте Oracle) |
| 4. | http://docs.oracle.com/en/operating-systems/ | Документация по продуктам Oracle, раздел Oracle Operating System |
| 5. | https://github.com/illumos/illumos-gate | Сайт проекта OpenIndiana (содержит исходные тексты OpenSolaris) |
| 6. | http://pubs.opengroup.org/onlinepubs/9699919799/ | Действующая редакция стандарта POSIX: The Open Group Base Specifications Issue 7, IEEE Std 1003.1™, 2013 Edition, |

### 7.2 Современные профессиональные базы данных.

Не используются.

## 8. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине.

### 8.1 Перечень программного обеспечения

Для обеспечения реализации дисциплины используется стандартный комплект программного обеспечения (ПО), включающий регулярно обновляемое лицензионное ПО Windows и MS Office.

Для обеспечения реализации дисциплины используется ОС OpenIndiana (бесплатное программное обеспечение).

### 8.2 Информационные справочные системы

Не используются

## 9. Материально-техническая база, необходимая для осуществления образовательного процесса по дисциплине.

Для реализации дисциплины используются специальные помещения:

1. Учебные аудитории для проведения занятий лекционного типа, практических занятий, курсового проектирования (выполнения курсовых работ), групповых и индивидуальных консультаций, текущего контроля успеваемости, промежуточной и итоговой аттестации.

2. Помещения для самостоятельной работы обучающихся.

Учебные аудитории укомплектованы специализированной мебелью и техническими средствами обучения, служащими для представления учебной информации большой аудитории.

Помещения для самостоятельной работы обучающихся оснащены компьютерной техникой с возможностью подключения к сети "Интернет" и обеспечением доступа в электронную информационно-образовательную среду НГУ.

Материально-техническое обеспечение образовательного процесса по дисциплине для обучающихся из числа лиц с ограниченными возможностями здоровья осуществляется согласно «Порядку организации и осуществления образовательной деятельности по образовательным программам для инвалидов и лиц с ограниченными возможностями здоровья в Новосибирском государственном университете».

## 10. Оценочные средства для проведения текущего контроля успеваемости и промежуточной аттестации по дисциплине.

### 10.1. Порядок проведения текущего контроля успеваемости и промежуточной аттестации по дисциплине

**Текущий контроль успеваемости**

Во время практических занятий студентам необходимо сдавать задания. Если в течение семестра сдано менее 6 заданий, студент не может получить итоговую оценку выше, чем «удовлетворительно». Если сдано от 6 до 11 заданий, то оценка не может быть выше, чем «хорошо». Если в течение семестра сдано 23 задания и более, студент имеет право на экзамене сдавать только один вопрос билета (по своему выбору) вместо двух. Примеры заданий приведены в п. 10.3.

**Промежуточная аттестация**

Освоение компетенций оценивается согласно шкале оценки уровня сформированности компетенции. Положительная оценка по дисциплине выставляется в том случае, если заявленная компетенция ОПК-3 сформирована не ниже порогового уровня.

Окончательная оценка работы студента в течение семестра происходит на экзамене. Экзамен проводится в конце семестра в экзаменационную сессию по билетам в устной форме с учетом результатов текущего контроля успеваемости. Вопросы билета подбираются таким образом, чтобы проверить уровень сформированности компетенции ОПК-3.
Вывод об уровне сформированности компетенций принимается преподавателем. Оценки «отлично», «хорошо», «удовлетворительно» означают успешное прохождение промежуточной аттестации.

**Соответствие индикаторов и результатов освоения дисциплины**

Таблица 10.1

| **Индикатор** | **Результат обучения по дисциплине** | **Оценочные средства** |
|---|---|---|
| **ОПК - 3.1.** Применяет различные источники информации для решения задач профессиональной сферы деятельности. | Знать теоретические основы, используемые при построении современных операционных систем; современное состояние компьютерных технологий, используемых для научных и практических целей. | Задания на лабораторных работах, экзамен в устной форме. |
| **ОПК – 3.2.** Применяет основные приемы, возможности и правила работы со стандартными и специализированными программными продуктами при решении профессиональных задач.<br><br>**ОПК – 3.3.** Применяет методологию поиска научной и технической информации в сети Интернет и специализированных базах данных. | **Уметь** выбирать программное обеспечение для решения конкретной проблемы; работать с программными интерфейсами системных вызовов и библиотек, соответствующих стандарту POSIX. | Задания на лабораторных работах, экзамен в устной форме. |

**10.2. Описание критериев и шкал оценивания индикаторов достижения результатов обучения по дисциплине «Операционные системы».**

Таблица 10.2

| Критерии оценивания результатов обучения | Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций) | Уровень освоения компетенции | | | |
|---|---|---|---|---|---|
| | | Не сформирован (0 баллов) | Пороговый уровень (3 балла) | Базовый уровень (4 балла) | Продвинутый уровень (5 баллов) |
| 1 | 2 | 3 | 4 | 5 | 6 |
| Полнота знаний | ОПК 3.1 | Уровень знаний ниже минимальных требований. Имеют место | Демонстрирует общие знания базовых понятий по | Уровень знаний соответствует программе подготовки по темам/разделам | Уровень знаний соответствует программе подготовки по темам/разделам |

| | | | | | |
|---|---|---|---|---|---|
| | | грубые ошибки. | темам/разделам дисциплины. Допускается значительное количество негрубых ошибок. | дисциплины. Допускается несколько негрубых/ несущественных ошибок. Не отвечает на дополнительные вопросы. | дисциплины. Свободно и аргументированно отвечает на дополнительные вопросы. |
| Наличие умений | ОПК 3.2<br>ОПК 3.3 | Отсутствие минимальных умений. Не умеет решать стандартные задачи. Имеют место грубые ошибки. | Продемонстрированы частично основные умения. Решены типовые задачи. Допущены негрубые ошибки. | Продемонстрированы все основные умения. Решены все основные задания с негрубыми ошибками или с недочетами. | Продемонстрированы все основные умения. Решены все основные задания в полном объеме без недочетов и ошибок. |

**10.3. Типовые контрольные задания и материалы, необходимые для оценки результатов обучения.**

**Задания на практических работах**

**Раздел 1**

## Среда исполнения

### 1. Вывод различных атрибутов процесса в соответствии с указанными опциями

Напишите программу, которая будет обрабатывать опции, приведенные ниже. Опции должны быть обработаны в соответствии с порядком своего появления справа налево. Одной и той же опции разрешено появляться несколько раз. Используйте getopt(3C) для определения имеющихся опций. Сначала пусть ваша программа обрабатывает только некоторые опции. Затем добавьте еще, до тех пор, пока все требуемые опции не будут обрабатываться. Вы можете скопировать воспользоваться программой getopt_ex.c и изменить ее.

- -i Печатает реальные и эффективные идентификаторы пользователя и группы.
- -s Процесс становится лидером группы. Подсказка: смотри setpgid(2).
- -p Печатает идентификаторы процесса, процесса-родителя и группы процессов.
- -u Печатает значение ulimit
- -Unew_ulimit Изменяет значение ulimit. Подсказка: смотри atol(3C) на странице руководства strtol(3C)
- -c Печатает размер в байтах core-файла, который может быть создан.
- -Csize Изменяет размер core-файла
- -d Печатает текущую рабочую директорию
- -v Распечатывает переменные среды и их значения
- -Vname=value Вносит новую переменную в среду или изменяет значение существующей переменной.

Проверьте вашу программу на различных списках аргументов, в том числе:

- Нет аргументов
- Недопустимую опцию.
- Опции, разделенные знаком минус.
- Неудачное значение для U.

### 2. Время в Калифорнии

Измените программу ex_time.c, чтобы она выводила дату и время в Калифорнии (Pacific Standard Time, PST). Подсказка: Если время UTC 20 часов, то в Калифорнии 12 часов.

### 3. Установка идентификатора пользователя для доступа к файлу

Создайте файл данных, который может писать и читать только владелец (это можно сделать командой shell chmod 600 file) и напишите программу, которая

1. Печатает реальный и эффективный идентификаторы пользователя.
2. Открывает файл с помощью fopen(3). Если fopen() завершился успешно, файл должен быть закрыт с помощью fclose(3). Напечатайте сообщение об ошибке, используя perror(3C), если файл не удалось открыть.
3. Сделайте, чтобы реальный и эффективный идентификаторы пользователя совпадали. Подсказка: setuid(2)
4. Повторите первые два шага.

Проверьте работу вашей программы.

5. Исполните программу и посмотрите вывод
6. Сделайте программу доступной для запуска членам вашей группы и пусть ваши одногруппники исполнят программу.
7. Командой chmod u+s prog установите бит установки идентификатора пользователя и пусть ваши одногруппники опять исполнят эту программу.

## Управление памятью

### 4. Список строк

Напишите программу, которая вставляет строки, введенные с клавиатуры, в список. Память под узлы списка выделяйте динамически с использованием malloc(3). Ввод завершается, когда в начале строки вводится точка (.). Затем все строки из списка выводятся на экран.
Подсказка: объявите массив символов размера, достаточного чтобы вместить самую длинную введенную строку. Используйте fgets(3), чтобы прочитать строку, и strlen(3), чтобы определить ее длину. Помните, что strlen(3) не считает нулевой символ, завершающий строку. После определения длины строки, выделите блок памяти нужного размера и внесите новый указатель в список.

## Системные вызовы ввода/вывода

### 5. Таблица поиска строк в текстовом файле.

Написать программу, которая анализирует текстовый файл, созданный текстовым редактором, таким как ed(1) или vi(1). После запроса, который предлагает ввести номер строки, с использованием printf(3) программа печатает соответствующую строку текста. Ввод нулевого номера завершает работу программы. Используйте open(2), read(2), lseek(2) и close(2) для ввода/вывода. Постройте таблицу отступов в файле и длин строк для каждой строки файла. Как только эта таблица построена, позиционируйтесь на начало заданной строки и прочтите точную длину строки.
Подсказка: Выберите или создайте текстовый файл с короткими строками. Помните, что первая строка начинается с нулевого отступа в файле. Найдите каждый символ перевода строки, запи-

шите его позицию; в программе следует использовать вызов lseek(fd, 0L, 1). Для отладки распечатайте эту таблицу и сравните с таблицей, полученной вручную. Как только таблицы начнут совпадать, можно приступать к запросу номера строки.

### 6. Таблица поиска строк в текстовом файле.

Измените программу так, чтобы пользователю отводилось 5 секунд на ввод номера строки. Если пользователь не успевает, программа должна распечатать все содержимое файла и завершиться. Если же пользователь успел в течение пяти секунд ввести номер строки, то программа должна работать как в предыдущей задаче.

### 7. Таблица поиска строк в текстовом файле 2.

Измените предыдущую программу так, чтобы использовалось отображение файла в память взамен использования read(2), lseek(2) и write(2).

## Захват файлов и записей

### 8. Защищенный текстовый редактор

Напишите программу, которая захватывает весь файл перед вызовом редактора. Это защитит файл с правами доступа группы на изменение. После того как программа захватит файл, вызовите свой любимый редактор с помощью библиотечной функции system(3). Попытайтесь в вашей программе использовать допустимое захватывание. Когда ваша программа захватит файл, попросите одногруппника исполнить вашу программу. Объясните результат. Потом попытайтесь использовать обязательное захватывание. Объясните результат.

## Создание процессов и исполнение программ. Управление процессами

### 9. Создание двух процессов

Напишите программу, которая создает подпроцесс. Этот подпроцесс должен исполнить cat(1) длинного файла. Родитель должен вызвать printf(3) и распечатать какой-либо текст. После выполнения первой части задания модифицируйте программу так, чтобы последняя строка, распечатанная родителем, выводилась после завершения порожденного процесса. Используйте wait(2), waitid(2) или waitpid(3).

### 10. Код завершения команды

Напишите программу, которая запускает команду, заданную в качестве первого аргумента, в виде порожденного процесса. Все остальные аргументы программы передаются этой команде. Затем программа должна дождаться завершения порожденного процесса и распечатать его код завершения.

### 11. Функция execvpe()

Напишите функцию execvpe(), которая работает как execvp(2), но позволяет изменять среду исполнения, как execve(2).
Совет: используйте внешнюю переменную environ.

### 12. Командный интерпретатор shell (задание №1 по shell)

В этом упражнении вы должны разработать упрощенную версию командного интерпретатора системы UNIX - shell. В дальнейшем этот shell будет улучшен добавлением программных каналов и обработки сигналов. Для вас заранее написаны некоторые функции этой программы. В

файле shell.c содержится скелет функции main() с комментариями, говорящими, где следует вставить ваш код. Функция main() вызывает функцию prompline() из файла promptline.c, которая читает строку из стандартного ввода. Затем main() вызывает parseline() из файла parseline.c, которая проверяет ввод на отсутствие синтаксических ошибок. Для каждой команды в строке, parseline() формирует дескриптор команды типа struct command, определенного в shell.h Существует глобальный массив таких структур, называемый cmds[] и объявленный в shell.c. Дескриптор команды состоит из двух полей. Первое поле - массив указателей, называемый cmdargs[]. Функция parseline() инициализирует этот массив указателями на строки - аргументы. Каждый аргумент завершается нулевым байтом. Во входной строке аргументы разделяются пробелами. Кроме того, parseline() помещает после последнего аргумента в списке cmdargs[] нулевой указатель. Команды могут разделяться переводом строки, точкой с запятой, амперсендом (&) или символами <, >, >> или |. Другие глобальные переменные, объявленные в shell.c определяют, должна ли команда быть запущена в основном (foreground) или фоновом (background) режиме, а также для перенаправления ввода/вывода из/в файл. Эти переменные (bkgrnd, infile, outfile и appfile) устанавливаются в функции parseline(). Если bkgrnd ненулевой, команда должна быть запущена в фоновом режиме, то есть ваш shell не должен ожидать завершения соответствующего процесса. Если infile ненулевой, он указывает на имя файла, из которого должен быть получен стандартный ввод первой команды в строке, т.е. символ <. Аналогично, если outfile ненулевой, он указывает на имя файла, куда следует перенаправить вывод последней команды в строке, т.е. символ >. Наконец, если appfile ненулевой, это имя файла, к содержимому которого следует добавить вывод последней команды в строке, т.е. символ >>. В следующих упражнениях вы будете модифицировать shell.c. Каждое последующее упражнение является улучшением предыдущего. Вы должны использовать следующую версию вызова exec(2):
execvp(cmds[i].cmdargs[0], cmds[i].cmdargs);

### 13. Исполнение команд (задание №2 по shell)

Модифицируйте shell.c для исполнения каждой команды в массиве cmds[] как подпроцесса. Во время исполнения этого подпроцесса, ваша программа должна ждать его завершения. Если программа, заданная во входной строке, не может быть найдена, shell должен распечатать сообщение об ошибке и продолжить исполнение. Предполагается, что пользователь набирает только простые команды, т.е. только одна команда на строке, без метасимволов, перенаправления ввода/вывода, конвейеров и символа &.

### 14. Исполнение в фоновом режиме (задание №3 по shell)

Измените вашу программу так, чтобы позволить исполнение команд в фоновом режиме, обозначаемое символом & в конце командной строки. Программа должна выводить идентификатор фонового процесса на стандартный вывод.

### 15. Перенаправление ввода/вывода (задание №4 по shell)

Измените вашу программу так, чтобы позволить перенаправление ввода/вывода. Заметьте, что перенаправление ввода ("< filename") используется только с первой командой в строке, а вывода ("> filename" или ">> filename") - только с последней.
Совет: используйте dup(2).

## Управление терминальным вводом/выводом

### 16. Ответ без ввода новой строки

Напишите программу, которая печатает вопрос и требует односимвольного ответа. Измените атрибуты вашего терминала так, чтобы пользователю не нужно было вводить новую строку после ответа.

### 17. Строчный редактор

Многие программы, принимающие ввод с терминала, позволяют редактировать строку перед использованием. Напишите программу, которая выключает эхо и каноническую обработку, таким образом выключив и обработку символа забоя. Ваша программа должна получать ввод с клавиатуры и показывать его на терминале в соответствии со следующими правилами:

8. Каждый введенный символ должен немедленно появляться на дисплее.
9. Когда вводится символ ERASE, стирается последний символ в текущей строке.
10. Когда вводится символ KILL, стираются все символы в текущей строке.
11. Когда вводится CTRL-W, стирается последнее слово в текущей строке, вместе со всеми следующими за ним пробелами.
12. Программа завершается, когда введен CTRL-D и курсор находится в начале строки.
13. Все непечатаемые символы, кроме перечисленных выше, должны издавать звуковой сигнал, выводя на терминал символ CTRL-G.
14. Длина строки ограничена 40 символами. Если какое-то слово пересекает 40-й столбец, это слово должно быть помещено в начало следующей строки.

## Управление файлами

### 18. Листинг каталога

Напишите программу - аналог команды ls -ld. Для каждого своего аргумента эта команда должна распечатывать:

- Биты состояния файла в воспринимаемой человеком форме:
- d если файл является каталогом
- - если файл является обычным файлом
- ? во всех остальных случаях
- Три группы символов, соответствующие правам доступа для хозяина, группы и всех остальных:
- r если файл доступен для чтения, иначе -
- w если файл доступен для записи, иначе -
- x если файл доступен для исполнения, иначе -
- Количество связей файла
- Имена собственника и группы файла (совет - используйте getpwuid и getgrgid).
- Если файл является обычным файлом, его размер. Иначе оставьте это поле пустым.
- Дату модификации файла (используйте ctime).
- Имя файла (если было задано имя с путем, нужно распечатать только имя).

Желательно, чтобы поля имели постоянную ширину, т.е. чтобы листинг имел вид таблицы. Совет - используйте printf.

## Управление каталогами

### 19. Шаблоны имен файлов

Напишите программу, которая приглашает пользователя ввести шаблон имени файла, аналогичный тому, который используется в shell. Синтаксис шаблона таков:

- * соответствует последовательности любых символов кроме /, имеющей любую длину; возможно - пустой последовательности.
- ? соответствует любому одному символу.
- / не может встречаться.
- любой другой символ соответствует самому себе.

Символы * и ? в шаблоне могут встречаться в любом количестве и в любом порядке.
Затем программа должна найти и распечатать имена всех файлов в текущем каталоге, соответствующих шаблону. Если таких файлов нет, программа должна распечатать сам шаблон.
Совет: используйте readdir, чтобы считать все имена файлов в текущем каталоге, и выберите из них соответствующие шаблону.

### 20. Шаблоны имен файлов (2)

Измените предыдущую программу так, чтобы в шаблоне могли встречаться символы /. При этом программа должна распечатывать все файлы, путевые имена которых соответствуют шаблону. Так, шаблону */* соответствуют все файлы во всех подкаталогах текущего каталога.

## Раздел 2

## Сигналы

### 21. Пищалка

Напишите программу, которая входит в бесконечный цикл и издает звуковой сигнал на вашем терминале каждый раз, когда вы вводите символ, на который у вас настроена посылка сигнала SIGINT (по умолчанию CTRL-C). При получении SIGQUIT, она должна вывести сообщение, говорящее, сколько раз прозвучал сигнал, и завершиться.

### 22. Мультиплексирование ввода

Напишите программу, которая читает из нескольких файлов по очереди, т.е. после чтения строки из одного файла, читается строка из следующего и т.д. Если в течение TIME_OUT секунд ничего не было прочитано, берется следующий файл.
Программа получает в качестве аргументов имена одного или нескольких файлов, из которых она будет читать. Обычно это терминальные файлы (т.е. /dev/ttynn), но могут быть файлы и других типов. (read(2) с нетерминального устройства может прочитать несколько строк, в зависимости от количества требуемых байтов и длины этих строк.) Если в одном из файлов достигнут конец файла, из него больше не читают. Когда конец файла достигнут во всех файлах, программа завершается. Проверьте вашу программу так:
$ multiplex /dev/tty 'tty'

### 23. Защита от сигналов, посылаемых с терминала (задание №5 по shell)

Измените исходный текст программы shell так, чтобы она не завершалась, когда вы генерируете сигнал SIGINT. Вместо этого должен завершаться процесс первого плана, а ваш shell должен немедленно выдавать приглашение. Кроме того, предохраните команды, исполняемые в фоновом режиме, от прерывания сигналами SIGINT и SIGQUIT. (Если вы не имеете собственной

версии этой программы, вы можете посмотреть исходные тексты заготовки в файлах shell.c, parseline.c, promptline.c и shell.h. Откомпилировав их, вы получите простой командный интерпретатор, способный выполнять программы в виде порожденных процессов, запускать фоновые процессы и перенаправлять ввод/вывод.)

### 24. Простое управление заданиями (задание №6 по shell)

Если вы реализовали защиту от сигналов, модифицируйте ваш интерпретатор shell, так чтобы обеспечить следующие возможности для управления заданиями: SIGTSTP, посланный с клавиатуры (CTRL-Z по умолчанию), должен заставить основную программу перейти в фоновый режим и возобновить исполнение. PID переведенного на фон процесса будет выведен на stderr, и командный интерпретатор выдаст приглашение для следующей команды.
Вам необходимо реализовать встроенные команды fg и bg, аналогичные одноименным командам bash(1) или ksh(1).
Совет: сделайте каждый порожденный процесс лидером группы процессов и явным образом переводите его на первый план. Когда порожденный процесс первого плана завершается или останавливается, переводите на первый план интерпретатор.

## Программные каналы

### 25. Связь через программный канал

Напишите программу, которая создает два подпроцесса, взаимодействующих через программный канал. Первый процесс выдает в канал текст, состоящий из символов верхнего и нижнего регистров. Второй процесс переводит все символы в верхний регистр, и выводит полученный текст на терминал. Подсказка: см. toupper(3).

### 26. Связь с использованием функций стандартной библиотеки

Используйте стандартные библиотечные функции popen(3) и pclose(3) для выполнения тех же операций, что и в предыдущем упражнении.

### 27. Подсчет пустых строк в файле

Напишите программу, которая подсчитывает пустые строки в файле, используя команду wc(1).

### 28. Генератор случайных чисел

Напишите программу, которая генерирует сортированный список из ста случайных чисел в диапазоне от 0 до 99. Распечатайте числа по десять в строке. Используйте p2open(3), чтобы запустить sort(1) и rand(3) и srand(3) для генерации случайных чисел.

### 29. Конвейеры (задание №7 по shell)

Измените ваш командный интерпретатор так, чтобы он позволял создавать конвейеры. Если вы добавили управление заданиями в упражнениях Раздела 1, вы можете модифицировать программу так, чтобы все процессы в конвейере принадлежали к одной группе. Тогда, например, SIGINT мог бы прервать все процессы в конвейере первого плана.

## Сокеты

### 30. Связь через Unix domain socket

Напишите две программы, взаимодействующих через Unix domain socket. Первый процесс (сервер) создает сокет и слушает на нем. При присоединении клиента, сервер получает через соединение текст, состоящий из символов верхнего и нижнего регистров, переводит его в верхний

регистр и выводит в свой стандартный поток вывода, аналогично задаче 25. Второй процесс (клиент) устанавливает соединение с сервером и передает ему текст. После разрыва соединения клиентом, оба процесса завершаются.

### 31. Связь нескольких процессов через Unix domain socket

Напишите две программы, взаимодействующих через Unix domain socket. Первый процесс (сервер) создает сокет и слушает на нем. При присоединении клиента, сервер получает через соединение текст, состоящий из символов верхнего и нижнего регистров, переводит его в верхний регистр и выводит в свой стандартный поток вывода, аналогично задаче 25. Второй процесс (клиент) устанавливает соединение с сервером и передает ему текст. Необходимо обеспечить возможность подключения нескольких клиентов и параллельное (без задержек) получение текста от них. При этом, преобразованный текст разных клиентов в выдаче сервера может смешиваться.
Для мультиплексирования соединений клиентов используйте select(3C) или poll(2).

### 32. Асинхронный ввод-вывод

Реализуйте задачу 31, используя асинхронный ввод-вывод вместо select(3C)/poll(2).
33. Прокси-сервер
Реализуйте сервер, который принимает TCP соединения и транслирует их. Сервер должен получать из командной строки следующие параметры:

15. Номер порта P, на котором следует слушать.
16. Имя или IP-адрес узла N, на который следует транслировать соединения.
17. Номер порта P', на который следует транслировать соединения.

Сервер принимает все входящие запросы на установление соединения на порт P. Для каждого такого соединения он открывает соединение с портом P' на сервере N. Затем он транслирует все данные, получаемые от клиента, серверу N, а все данные, получаемые от сервера N – клиенту. Если сервер N или клиент разрывают соединение, наш сервер также должен разорвать соединение. Если сервер N отказывает в установлении соединения, следует разорвать клиентское соединение.
Сервер должен обеспечивать трансляцию 510 соединений при лимите количества открытых файлов на процесс 1024. Сервер не должен быть многопоточным и никогда не должен блокироваться при операциях чтения и записи. Не следует использовать неблокирующиеся сокеты. Следует использовать select(3C) или poll(2).

### 34-35. Проброс порта

Эта задача засчитывается за две. Реализуйте две программы, приемник и передатчик. Передатчик слушает на некотором порту TCP. Все соединения с этим портом транслируются на приемник. Приемник, в свою очередь, транслирует данные всех соединений на сервер: некоторый другой сервис, слушающий на другом порту и, возможно, на другой машине. Таким образом, клиенту (процессу, присоединяющемуся к приемнику) создается иллюзия, что он присоединился непосредственно к серверу.
Например, если приемнику указать в качестве сервиса localhost:22, это можно использовать для туннелирования соединений ssh. В свою очередь, сама пара приемник-передатчик аналогична возможности “проброса портов” в протоколе ssh. Изучите проброс портов протокола ssh, если вы не понимаете задачу.
Между приемником и передатчиком разрешается установить только одно соединение через сокет TCP. Все данные транслируемых соединений передаются через единственный последовательный канал.
Для определения того, к какому соединению относятся какие данные, следует использовать специальный протокол. Данные в соединении приемник-передатчик (туннеле) разбиты на кадры. Каждый кадр содержит данные одного соединения. Кадр начинается с байта 01111110b (0x7E). Следующий байт содержит номер соединения (таким образом, ваш туннель не может поддерживать более 255 соединений одновременно). Затем идут данные соединения. Кадр

также заканчивается байтом 01111110b. Такая схема кодирования теоретически допускает кадры неограниченной длины, но ваша реализация может накладывать какое-то ограничение на размер кадра, возможно, связанное с размером буфера.
Если в потоке данных клиентского соединения встречается байт 01111110b, его следует заменить на пару байтов 01111101b 01111110b, чтобы получатель не проинтерпретировал этот байт как конец кадра. В свою очередь, байт 01111101b следует заменить на пару байтов 01111101b 01111101b. Перед передачей данных клиенту или серверу следует произвести обратную замену.
Если вы не понимаете, почему это обеспечивает однозначное декодирование передаваемых данных, вспомните задачу RLE из курса “Цифровые платформы” либо то, каким образом кодируются двойная кавычка и обратный слэш в строковых литералах языка C, либо прочитайте описание формата кадров протокола HDLC.
Номер соединения 0 следует использовать для канала управления: запросов на установление нового соединения или разрыв существующего. Протокол канала управления разработайте самостоятельно. Обратите внимание, что запрос на установление соединения может посылать только приемник, но разрыв соединения может происходить как по инициативе клиента, так и по инициативе сервера.
Рекомендации: для мультиплексирования соединений, используйте select(3C) или poll(2). Не допускается использование неблокирующихся сокетов и холостых циклов. Избегайте чтения и записи данных (вызовов read/write или send/receive) по одному байту. При перекодировании данных, учтите, что их объем может измениться.
Для демонстрации приложения можно использовать туннелирование протокола ssh или http proxy.

### 36. Псевдомногопоточный HTTP-клиент

Реализуйте простой HTTP-клиент. Он принимает один параметр командной строки – URL. Клиент делает запрос по указанному URL и выдает тело ответа на терминал как текст (т.е. если в ответе HTML, то распечатывает его исходный текст без форматирования). Вывод производится по мере того, как данные поступают из HTTP-соединения. Когда будет выведено более экрана (более 25 строк) данных, клиент должен продолжить прием данных, но должен остановить вывод и выдать приглашение Press space to scroll down.
При нажатии пользователем клиент должен вывести следующий экран данных. Для одновременного считывания данных с терминала и из сетевого соединения используйте системный вызов select или poll.
37. Псевдомногопоточный HTTP-клиент 2
Реализуйте задачу упр. 36, используя системные вызовы aio_read/aio_write.

## Список билетов на экзамен

Билет I.

1. Как происходит загрузка операционной системы? Что такое первичный загрузчик? Вторичный? Как происходит загрузка бездисковых машин?

2. Организация файловой системы HPFS.

Билет II.

1. Распределение памяти алгоритмами близнецов и парных меток. Ограничения этих алгоритмов.

2. Аутентификация и проверка подлинности кода в Apple iOS.

Билет III.

1. Алгоритмы поиска жертвы при страничном обмене и кэшировании. Критерии выбора и влияние алгоритма на производительность. Что такое рабочее множество страниц?

2. Сигналы в системах семейства Unix.

Билет IV.

1. Инверсия приоритета. Способы ее предотвращения и способы обхода этой проблемы.

2. Линки в транспьютере.

Билет V.

1. Определение задачи реального времени. Чем системы РВ отличаются от систем разделенного времени? Пример архитектуры ОС реального времени.

2. Сборщик мусора Java HotSpot.

Билет VI.

1. Журнальные файловые системы. Принципы работы. Для чего этонужно?

2. Почтовые ящики (mailbox) в VAX/VMS.

Билет VII.

1. Семафоры Дийкстры. Мутексы, двоичные семафоры и семафоры общего вида. Мертвая блокировка и способы избежать ее.

2. Файловая система NetApp WAFL.

Билет VIII.

1. Как реализуется многопоточность на однопроцессорной машине. Что такое контекст процесса? Какие особенности процессора влияют на скорость переключения процессов?

2. Формирование запросов на ввод/вывод в RSX-11, VMS, OpenVMS. Какие преимущества предоставляет этот метод?

Билет IX.

1. Что такое гармонически взаимодействующие последовательные процессы? Средства для реализации этой дисциплины в существующих системах.

2. Организация страничного обмена в VMS, OpenVMS и Windows NT.

Билет X.

1. Методы реализации виртуальной памяти. Базовая адресация, сег-ментная и страничная виртуальная память.

2. Программные каналы (трубы) в системах семейства Unix.

Билет XI.

1. Что такое абсолютный и относительный загрузчики? Структура абсолютного и перемещаемого загрузочных модулей. Что такое позиционно-независимый код?

2. Семафоры Unix System V IPC. Наборы семафоров.

Билет XII.

1. Устойчивые к сбоям файловые системы. Методы реализации устойчивых ФС.

2. Диспетчер задач в транспьютере.

Билет XIII.

1. Сборка в момент загрузки. Преимущества и недостатки этого метода. Чем отличаются DLL Win32 и разделяемые библиотеки ELF.

2.Динамическое выделение памятив ОС семейства Unix и стандарте POSIX.

Билет XIV.

1. Драйвер устройства. Функции драйвера в ОС семейства Unix.

2. Файловая система FAT.

Билет XV.

1. Динамическое выделение памяти. Методы борьбы с фрагментацией. Основные алгоритмы выделения памяти.

2. Флаги событий в RSX и VMS. Что такое AST?.

Билет XVI.

1. Мертвая и живая блокировки. Способы их предотвращения. Преимущества и недостатки каждого из методов

2. Разделяемые библиотеки формата ELF.

Билет XVII.1. Разделяемая память. Преимущества и недостатки по сравнению с другими методами межпроцессного взаимодействия.

2. Механизм setuid в ОС семейства Unix.

Билет XVIII.

1. Событийно-ориентированные системы. Обязательно ли такая система является многопоточной?

2. Понятия инода и связи в файловых системах ОС семейства Unix.

Билет XIX.1. Реентерабельная программа. Техника реализации реентерабельных программ. Всегда ли это возможно? Что такое критическая секция?

2. Загружаемые модули и разделяемые библиотеки Win32/Win64(PE).

Билет XX.

1.Прерывания в классических процессорах (PDP-11,8086, x86). Внешние прерывания и исключения (exceptions).

2. Сборщик мусора Java G1.

Билет XXI.

1.Объектный модуль. Объектная библиотека. Структуры данных, содержащиеся в объектном модуле, в общих чертах. Алгоритм работы сборщика и выбора модулей из архивной библиотеки.

2. Структура и принципы работы файловой системы NTFS.

Билет XXII.

1.Приоритеты процессов и нитей. Управление приоритетами для нитей реального и разделенного времени. Где используется и для чего нужно динамическое изменение приоритета?

2. Права доступа к файлам в ОС семейства Unix.

Билет XXIII.

1. Системы управления доступом. Полномочия и списки контроля доступа. Кольца доступа.

2. Запуск задач в ОС семейства Unix.

Билет XXIV.

1. Планировщики разделенного времени. Динамическое управление приоритетами в системах разделенного времени.

2. Структура и особенности организации файловой системы UFS(FFS).

Билет XXV.

1. Кооперативная и вытесняющая (preemptive) многозадачность. Преимущества и недостатки обоих архитектур.

2. Файловая система ISO 9660 (CDFS).

Билет XXVI.

1. Троянские программы и способы их внедрения. Меры по защите от троянских программ.

2. Асинхронный ввод-вывод в стандарте POSIX.

Билет XXVII.

1. Сборка мусора. Основные стратегии сборки мусора, их преимущества и недостатки

2. Загрузка ОС на PC-совместимых компьютерах.

Билет XXVIII.

1. Ввод-вывод в режиме опроса и по прерываниям. Преимущества и недостатки.

2. Уровни RAID.

Билет XXIX.1. Спинлоки и их применение. Их преимущества и недостатки по сравнению с другими средствами взаимоисключения.

2. Структура файловой системы RT-11.

Оценочные материалы по промежуточной аттестации, предназначенные для проверки соответствия уровня подготовки по дисциплине требованиям СУОС, хранятся на кафедре-разработчике РПД в печатном и электронном виде.

**Лист актуализации рабочей программы**
**по дисциплине «Операционные системы»**
**направление подготовки: 03.03.02 Физика**
**Направленность (профиль): Физическая информатика**

| № | Характеристика внесенных изменений (с указанием пунктов документа) | Дата и № протокола Учёного совета ФФ НГУ | Подпись ответственного |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |