

**Федеральное государственное автономное образовательное
учреждение высшего образования
«Московский физико-технический институт
(национальный исследовательский университет)»**

УТВЕРЖДЕНО

**Проректор по учебной работе и
довузовской подготовке**

А.А. Воронов

	Рабочая программа дисциплины (модуля)
по дисциплине:	Операционные системы
по направлению:	Прикладные математика и физика
профиль подготовки:	Беспилотные авиационные системы Физтех-школа авиационных и цифровых технологий кафедра информатики и вычислительной математики
курс:	2
квалификация:	бакалавр

Семестр, формы промежуточной аттестации: 3 (осенний) - Дифференцированный зачет

Аудиторных часов: 60 всего, в том числе:

лекции: 30 час.

семинары: 0 час.

лабораторные занятия: 30 час.

Самостоятельная работа: 75 час.

Всего часов: 135, всего зач. ед.: 3

Количество контрольных работ, заданий: 4

Программу составили:

Н.И. Хохлов, канд. физ.-мат. наук, заведующий кафедрой

В.Е. Карпов, канд. физ.-мат. наук, доцент, доцент

К.А. Коньков, канд. физ.-мат. наук, доцент, доцент

Программа обсуждена на заседании кафедры информатики и вычислительной математики 29.05.2020

Аннотация

Студенты получают базовые знания об организации операционных систем, разделении обязанностей между аппаратным обеспечением и ядром операционной системы. В курсе будет проведено рассмотрение концепций современных операционных систем производится на примере операционной системы Unix. Будет проводиться практика по пользовательскому интерфейсу Unix, программированию на языке Unix Shell, с использованием системных вызовов для взаимодействия с ядром в программах на языке Си. Студентам будет предоставлено две контрольные работы для проверки полученных знаний.

1. Цели и задачи

Цель дисциплины

- получение базовых знаний об организации операционных систем, разделении обязанностей между аппаратным обеспечением и ядром операционной системы. Рассмотрение концепций современных операционных систем производится на примере операционной системы Unix. Рассматриваются пользовательский интерфейс Unix, программирование на языке Unix Shell, использование системных вызовов для взаимодействия с ядром в программах на языке Си.

Задачи дисциплины

- изучение основных концепций и принципов проектирования операционных систем. Рассмотрение взаимодействия ядра операционной системы с аппаратным обеспечением современных компьютеров.
- рассмотрение реализации основных концепций современных ОС на примере Unix (понятия процесс, планировщик процессов файл и др.)
- знакомство с командной оболочкой Unix Shell на уровне пользователя и программиста. Выполнение лабораторных работ по написанию Shell-скриптов. Выполнение лабораторных работ на других скриптовых языках, в том числе, sed и AWK.
- изучение основных системных вызовов Unix. Программирование на языке Си с использованием системных вызовов.

2. Перечень формируемых компетенций

Освоение дисциплины направлено на формирование следующих компетенций:

Код и наименование компетенции	Индикаторы достижения компетенции
ОПК-1 Способен применять фундаментальные знания, полученные в области физико-математических и (или) естественных наук, и использовать их в профессиональной деятельности	ОПК-1.1 Способен анализировать поставленную задачу, намечать пути ее решения
	ОПК-1.2 Способен строить математические модели, производить количественные расчеты и оценки
	ОПК-1.3 Способен определять границы применимости полученных результатов
ПК-2 Способен анализировать полученные в ходе научно-исследовательской работы данные и делать научные выводы (заключения)	ПК-2.1 Владеет методами статистической обработки и анализа научных данных

3. Перечень планируемых результатов обучения по дисциплине (модулю)

В результате освоения дисциплины обучающиеся должны

знать:

- основные компоненты ОС общего назначения, необходимые для её функционирования.
- основные команды, необходимые для уверенной работы в Unix Shell на уровне пользователя.
- управляющие операторы и управляющие конструкции Unix Shell, необходимые для написания shell-скриптов.

уметь:

- работать в командной оболочке Unix Shell, писать скрипты для Unix Shell, писать программы на языке Си с использованием системных вызовов ОС Unix.

владеть:

- приёмами программирования на скриптовых языках на примере Unix Shell, awk и sed.

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

4.1. Разделы дисциплины (модуля) и трудоемкости по видам учебных занятий

№	Тема (раздел) дисциплины	Трудоемкость по видам учебных занятий, включая самостоятельную работу, час.			
		Лекции	Семинары	Лаборат. работы	Самост. работа
1	Введение	6		6	10
2	Процессы и их планирование в операционной системе	6		6	12
3	Кооперация процессов	6		8	12
4	Управление памятью	2		2	8
5	Контрольная работа 1			2	2
6	Файловые системы	2		2	8
7	Система управления вводом-выводом	2		2	8
8	Сети и сетевые операционные системы	2		2	8
9	Проблемы безопасности операционных систем	2			5
10	Контрольная работа 2	2			2
Итого часов		30		30	75
Подготовка к экзамену		0 час.			
Общая трудоёмкость		135 час., 3 зач.ед.			

4.2. Содержание дисциплины (модуля), структурированное по темам (разделам)

Семестр: 3 (Осенний)

1. Введение

Цели и задачи курса. Понятие о вычислительном комплексе. Системное программное обеспечение и операционные системы. Краткая история эволюции вычислительных систем. Взаимное влияние software и hardware. Автономные, сетевые и распределенные операционные системы. Классификация автономных операционных систем по их назначению и структуре. Архитектурная классификация параллельных и распределенных операционных систем. Знакомство с операционной системой UNIX. Системные вызовы и библиотека libc. Понятия login и password. Упрощенное устройство файловой системы в UNIX. Полные имена файлов. Текущая директория. Относительные имена файлов. Домашняя директория пользователя. Команда man – универсальный справочник. Команды cd и ls. Перенаправление стандартного ввода и стандартного вывода. Простейшие команды работы с файлами – cat, cp, mkdir, mv, rm. Шаблоны имен файлов. Пользователь и группа. Системные вызовы getuid() и getgid(). Команды chown и chgrp. Права доступа к регулярному файлу и к директории. Команда chmod. Маска создания файлов. Команда umask. Редактирование файлов, компиляция и запуск программ.

2. Процессы и их планирование в операционной системе

Понятие процесса. Процесс и программа. Состояния процесса. Управляющий блок процесса и его контекст. Операции над процессами. Переключение контекста. Уровни планирования процессов. Критерии планирования и требования к алгоритмам планирования. Параметры планирования. Вытесняющее и невытесняющее планирование. Алгоритмы планирования: FCFS, RR, SJF, гарантированное планирование, приоритетное планирование, многоуровневые очереди, многоуровневые очереди с обратной связью. Планирование в многопроцессорных и распределенных средах.

Понятие процесса в UNIX, его контекст. Идентификация процесса. Краткая диаграмма состояний процессов в UNIX. Иерархия процессов. Системные вызовы `getpid()` и `getppid()`. Создание процесса в UNIX. Системный вызов `fork()`. Завершение процесса. Функция `exit()`. Параметры функции `main()` в языке C. Переменные среды и аргументы командной строки. Изменение пользовательского контекста процесса. Семейство функций для системного вызова `exec()`.

3. Кооперация процессов

Взаимодействующие и независимые процессы. Категории средств связи. Установление и завершение связи. Прямая и косвенная адресация. Информационная валентность процессов и средств коммуникации. Симплексная, дуплексная и полудуплексная связь. Потoki ввода вывода и сообщения. Буферизация данных. Надежность обмена информацией. Нити исполнения и их отличие от процессов. `Interleaving`, `race condition` и взаимoisключения. Условия Бернштейна. Понятие критической секции процесса. Программные алгоритмы организации взаимодействия процессов и предъявляемые к ним требования. Семафоры, мониторы Хора и сообщения. Оптимистические стратегии в задачах синхронизации процессов и потоков.

Понятие потока ввода-вывода в операционной системе UNIX. Работа с файлами через системные вызовы и через функции стандартной библиотеки. Файловый дескриптор. Наследование файловых дескрипторов при системных вызовах `fork()` и `exec()`. Системные вызовы `open()`, `read()`, `write()`, `close()`. FIFO и pipe. Системные вызовы `pipe()`, `mknod()`, функция `mkfifo()`. Особенности системных потоковых вызовов при работе с FIFO и pipe. Преимущества и недостатки потокового обмена данными. IPC в UNIX. Пространство имен. Адресация в System V IPC. Функция `ftok()`. Дескрипторы System V IPC. Разделяемая память. Системные вызовы `shmget()`, `shmat()`, `shmdt()`, `shmctl()`. Команды `ipcs` и `ipcrm`. Нить исполнения (thread) в UNIX, ее идентификатор. Функция `pthread_self()`. Создание и завершение нити исполнения. Функции `pthread_create()`, `pthread_exit()`, `pthread_join()`. Семафоры в UNIX. Отличие операций над UNIX семафорами от классических операций. Системные вызовы `semget()`, `semop()`, `semctl()`. Понятие о POSIX семафорах. Очереди сообщений в UNIX. Системные вызовы `msgget()`, `msgsnd()`, `msgrcv()`, `msgctl()`. Понятие мультиплексирования. Мультиплексирование сообщений. Модель взаимодействия процессов клиент–сервер. Неравноправность клиента и сервера.

4. Управление памятью

Связывание адресов. Простейшие схемы управления памятью: схема с фиксированными разделами, своппинг, схема с переменными разделами. Проблема размещения больших программ. Понятие виртуальной памяти. Страничная память. Сегментная и сегментно-страничная организации памяти. Таблица страниц. Ассоциативная память. Иерархия памяти. Размер страницы. Исключительные ситуации при работе с памятью. Стратегии управления страничной памятью: выборки, размещения и замещения страниц. Алгоритмы замещения страниц: FIFO, OPT, LRU и другие. Трэшинг (thrashing). Свойство локальности. Модель рабочего множества. Функционирование менеджера памяти ОС Linux и Windows.

5. Контрольная работа 1

6. Файловые системы

Имена, структура, типы и атрибуты файлов. Операции над файлами. Директории. Операции над директориями. Защита файлов. Методы выделения дискового пространства: непрерывная последовательность блоков, связный список, связный список с индексацией, индексные узлы. Управление свободным и занятым дисковым пространством: битовый вектор, связный список. Асинхронный доступ к файлам. Проблемы надежности и производительности файловых систем.

Разделы носителя информации (partitions) в UNIX. Логическая структура файловой системы и типы файлов в UNIX. Организация файла на диске в UNIX на примере файловой системы s5fs. Понятие индексного узла (inode). Организация директорий (каталогов) в UNIX. Понятие суперблока. Указатель текущей позиции в файле. Системная таблица файлов и таблица индексных узлов открытых файлов. Операции над файлами и директориями. Понятие жестких и мягких связей. Системные вызовы и команды для выполнения операций над файлами и директориями: `chmod`, `chown`, `chgrp`, `open()`, `creat()`, `read()`, `write()`, `close()`, `stat()`, `fstat()`, `lstat()`, `ftruncate()`, `lseek()`, `link()`, `symlink()`, `unlink()`. Функции для изучения содержимого директорий `opendir()`, `readdir()`, `rewinddir()`, `closedir()`. Понятие о файлах, отображаемых в память (memory mapped файлах). Системные вызовы `mmap()`, `munmap()`. Понятие виртуальной файловой системы. Монтирование файловых систем в UNIX.

7. Система управления вводом-выводом

Общие сведения об архитектуре компьютера. Структура контроллера устройства. Опрос устройств и прерывания. Исключительные ситуации и системные вызовы. Прямой доступ к памяти (Direct Memory Access – DMA). Структура системы ввода-вывода. Систематизация внешних устройств и интерфейс между базовой подсистемой ввода-вывода и драйверами. Функции базовой подсистемы ввода-вывода. Блокирующиеся, не блокирующиеся и асинхронные системные вызовы. Буферизация и кэширование. Spooling и захват устройств. Обработка прерываний и ошибок. Планирование запросов. Алгоритмы планирования запросов к жесткому диску: FCFS, SSTF, SCAN, C-SCAN, LOOK, C-LOOK. Блочные и символьные устройства в UNIX. Понятие драйвера. Блочные, символьные драйверы, драйверы низкого уровня. Файловый интерфейс к драйверам. Коммутатор устройств. Старший и младший номер устройства. Понятие сигнала в UNIX. Способы возникновения сигналов и виды их обработки. Понятия группы процессов, сеанса, лидера группы, лидера сеанса, управляющего терминала сеанса, текущей и фоновой групп процессов. Системные вызовы `getpgrp()`, `setpgrp()`, `getpgid()`, `setpgid()`, `getsid()`, `setsid()`. Системный вызов `kill()` и команда `kill()`. Особенности получения терминальных сигналов текущей и фоновой группой процессов. Получение сигнала `SIGHUP` процессами при завершении лидера сеанса. Системный вызов `signal()`. Установка собственного обработчика сигнала. Сигналы `SIGUSR1` и `SIGUSR2`. Использование сигналов для синхронизации процессов. Завершение порожденного процесса. Системный вызов `waitpid()`. Сигнал `SIGCHLD` и его игнорирование. Возникновение сигнала `SIGPIPE` при попытке записи в `pipe` или `FIFO`, который никто не собирается читать. Понятие о надежности сигналов. POSIX функции для работы с сигналами.

8. Сети и сетевые операционные системы

Причины объединения компьютеров в сети. Сетевые и распределенные операционные системы. Взаимодействие удаленных процессов как основа работы вычислительных сетей. Многоуровневая модель построения сетевых вычислительных систем. Семейства и стеки протоколов. Эталонная модель OSI/ISO. Удаленная адресация и разрешение адресов. Понятие о DNS. Локальная адресация. Понятие порта. Полные адреса. Понятие сокета (socket). Фиксированная, виртуальная и динамическая маршрутизация. Связь с установлением логического соединения и передача данных с помощью сообщений.

Краткая история семейства протоколов TCP/IP. Общие сведения об архитектуре семейства протоколов TCP/IP. Уровень сетевого интерфейса. Уровень Internet. Протоколы IP, ICMP, ARP, RARP. Internet-адреса. Транспортный уровень. Протоколы TCP и UDP. Понятие порта. Понятие encapsulation. Уровень приложений/процессов. Использование модели клиент–сервер для взаимодействия удаленных процессов. Понятие socket в UNIX. Организация связи между удаленными процессами с помощью датаграмм. Организация связи между процессами с помощью установки логического соединения. Сетевой порядок байт. Функции `htons()`, `htonl()`, `ntohs()`, `ntohl()`. Функции преобразования IP-адресов `inet_ntoa()`, `inet_aton()`. Функция `bzero()`. Системные вызовы `socket()`, `bind()`, `sendto()`, `recvfrom()`, `accept()`, `listen()`, `connect()`.

9. Проблемы безопасности операционных систем

Классификация угроз. Формализация подхода к обеспечению информационной безопасности. Классы безопасности. Политика безопасности. Криптография как одна из базовых технологий безопасности ОС. Шифрование с симметричными и ассиметричными ключами. Правило Кирхгофа. Алгоритм RSA. Идентификация и аутентификация. Пароли, уязвимость паролей. Авторизация. Разграничение доступа к объектам ОС. Домены безопасности. Матрица доступа. Недопустимость повторного использования объектов. Аудит, учет использования системы защиты. Организация защищенных каналов и проблемы безопасности распределенных операционных систем.

10. Контрольная работа 2

5. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине (модулю)

- персональные компьютеры;
- учебная аудитория;
- мультимедийный проектор;
- экран.

6. Перечень рекомендуемой литературы

Основная литература

1. Основы операционных систем [Текст] : Курс лекций : учеб. пособие для вузов / В. Е. Карпов, К. А. Коньков .— 2-е изд., доп. и испр. — М. : Интернет - Ун-т информац. технологий, 2009, 2011 .— 536 с.
2. Операционная система UNIX [Текст] : учеб. пособие для вузов / А. М. Робачевский, С. А. Немнюгин, О. Л. Стесик .— 2-е изд., перераб. и доп. — СПб. : БХВ-Петербург, 2005, 2007, 2010 .— 656 с.

Дополнительная литература

1. Современные операционные системы [Текст] : [учеб. пособие для вузов] / Э. Таненбаум ; [пер. с англ. Н. Вильчинский, А. Лашкевич] .— 3-е изд. — СПб. : Питер, 2015 .— 1120 с

7. Перечень ресурсов информационно-телекоммуникационной сети "Интернет", необходимых для освоения дисциплины (модуля)

1. <http://www.citforum.ru/>
2. <http://www.opennet.ru/man.shtml> (оригиналы и переводы справочных руководств Unix).
3. <https://www.freebsd.org/cgi/man.cgi> (оригинальные MAN-страницы ОС FreeBSD)

8. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), включая перечень необходимого программного обеспечения и информационных справочных систем (при необходимости)

- удалённый Unix-сервер с помощью подключения по протоколу SSH
- Dev-C++ —интегрированная среда разработки приложений.

9. Методические указания для обучающихся по освоению дисциплины (модуля)

- чтение и конспектирование рекомендованной литературы;
- проработку учебного материала (по конспектам лекций, учебной и научной литературе), подготовку ответов на вопросы, предназначенных для самостоятельного изучения;

- доказательство отдельных утверждений, свойств;
- решение задач, предлагаемых студентам на лекциях;
- подготовку к дифференцированному зачету

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

по направлению:	Прикладные математика и физика
профиль подготовки:	Беспилотные авиационные системы Физтех-школа авиационных и цифровых технологий кафедра информатики и вычислительной математики
курс:	<u>2</u>
квалификация:	бакалавр

Семестр, формы промежуточной аттестации: 3 (осенний) - Дифференцированный зачет

Разработчики:

Н.И. Хохлов, канд. физ.-мат. наук, заведующий кафедрой

В.Е. Карпов, канд. физ.-мат. наук, доцент, доцент

К.А. Коньков, канд. физ.-мат. наук, доцент, доцент

1. Компетенции, формируемые в процессе изучения дисциплины

Код и наименование компетенции	Индикаторы достижения компетенции
ОПК-1 Способен применять фундаментальные знания, полученные в области физико-математических и (или) естественных наук, и использовать их в профессиональной деятельности	ОПК-1.1 Способен анализировать поставленную задачу, намечать пути ее решения
	ОПК-1.2 Способен строить математические модели, производить количественные расчеты и оценки
	ОПК-1.3 Способен определять границы применимости полученных результатов
ПК-2 Способен анализировать полученные в ходе научно-исследовательской работы данные и делать научные выводы (заключения)	ПК-2.1 Владеет методами статистической обработки и анализа научных данных

2. Показатели оценивания компетенций

В результате изучения дисциплины «Операционные системы» обучающийся должен:

знать:

- основные компоненты ОС общего назначения, необходимые для её функционирования.
- основные команды, необходимые для уверенной работы в Unix Shell на уровне пользователя.
- управляющие операторы и управляющие конструкции Unix Shell, необходимые для написания shell-скриптов.

уметь:

- работать в командной оболочке Unix Shell, писать скрипты для Unix Shell, писать программы на языке Си с использованием системных вызовов ОС Unix.

владеть:

- приёмами программирования на скриптовых языках на примере Unix Shell, awk и sed.

3. Перечень типовых (примерных) вопросов, заданий, тем для подготовки к текущему контролю

С целью контроля освоения обучающимися учебного материала проводится устный опрос в начале занятия или в конце занятия по пройденной теме.

3. Перечень типовых заданий, используемых для оценки знаний, умений, навыков Пример задач для контрольных работ

3 (осенний) семестр

1. Вход в систему Linux. Смена пароля. Использование команд интерпретатора *pwd*, *ls*, *cd*. Освоение встроенного справочника *man*. Использование команд *cat*, *cp*, *mkdir*, *mv*, *rm*, *chown*, *chgrp*, *chmod*, *umask*. Освоение программы Midnight Commander. Написание, отладка и запуск программы, выводящей идентификатор пользователя и его группы.
2. Прогон и анализ программы, порождающий новый процесс. Написание, отладка и прогон программы с разным поведением ребёнка и родителя. Написание, отладка и прогон программы, использующей аргументы функции *main()*. Прогон и анализ программы, изменяющей свой пользовательский контекст. Написание, отладка и прогон программы, порождающей новый процесс, который изменяет свой пользовательский контекст.
3. Прогон и анализ программы, записывающей информацию в файл в потоковом режиме. Написание, отладка и прогон программы, считывающей информацию из файла в потоковом режиме. Прогон и анализ программы, организующей *pipe* в одном процессе. Прогон и анализ программы, осуществляющей одностороннюю передачу данных через *pipe* между родителем и ребёнком. Написание, отладка и прогон программы, осуществляющей передачу данных через *pipe* между родителем и ребёнком в двух направлениях. Написание, отладка и прогон программы, определяющей ёмкость буфера *pipe*. Прогон и анализ программы, осуществляющей одностороннюю передачу данных через FIFO между родителем и ребёнком. Написание, отладка и прогон программы, осуществляющей передачу данных через FIFO между не близкородственными процессами.
4. Прогон и анализ двух программ, взаимодействующих через общую память и печатающих количество собственных запусков. Написание, отладка и прогон двух программ, одна из которых записывает в общую память свой исходный текст, а вторая – распечатывает его из общей памяти. Обнаружение *race condition* в предыдущих программах. Модернизация программ, считающих количество своих запусков, с помощью алгоритма Петерсона для корректного взаимодействия. Прогон и анализ программы, создающей новую нить исполнения. Написание, отладка и прогон программы, создающей несколько новых нитей исполнения. Написание, отладка и прогон программы, создающей несколько новых нитей исполнения с синхронизацией их работы.
5. Прогон и анализ двух программ, взаимодействующих через семафор (одна программа блокируется, если *n* раз не была запущена другая программа). Написание, отладка и прогон программы, порождающей новый процесс и осуществляющей полудуплексную связь между родителем и ребёнком через единственный *pipe* с синхронизацией с помощью семафора(ов). Модернизация программ, считающих количество своих запусков, с помощью семафоров для корректного взаимодействия.
6. Прогон и анализ двух программ, передающих текстовую информацию в одном направлении через очередь сообщений. Написание, отладка и прогон двух программ, передающих текстовую информацию в двух направлениях через единственную очередь сообщений. Написание, отладка и прогон двух программ, передающих смешанную информацию в двух направлениях через единственную очередь сообщений. Написание, отладка и прогон двух программ, осуществляющих взаимодействие клиент-сервер через единственную очередь сообщений.
7. Написание, отладка и прогон программы для определения глубины рекурсии символьной связи. Написание, отладка и прогон программы – аналога команды *ls*. Написание, отладка и прогон двух программ, взаимодействующих через файл, отображаемый в память.
8. Прогон и анализ программы, игнорирующей сигнал *SIGINT*. Написание, отладка и прогон программы, игнорирующей сигналы *SIGINT* и *SIGQUIT*. Прогон и анализ

программы, восстанавливающей предыдущую реакцию на сигнал SIGINT. Написание, отладка и прогон программы, восстанавливающей предыдущую реакцию на сигналы SIGINT и SIGQUIT. Написание, отладка и прогон двух программ, обменивающихся числовой информацией через сигналы.

9. Прогон, анализ и модификация сетевых программ клиент-сервер, взаимодействующих через протокол UDP. Прогон, анализ и модификация сетевых программ клиент-сервер, взаимодействующих через протокол TCP.

Примеры домашних заданий

3 (осенний) семестр

Задание 1

1. Напишите программу *useless* (*UNIX SYSTEM EXTREMELY LATE EXECUTION SOFTWARE SYSTEM*), которая читает файл и запускает указанные в нем программы с указанной задержкой от времени старта программы *useless*. Формат записи в файле:

<время задержки в секундах> <программа для выполнения>

Файл не является упорядоченным по временам задержки.

2. Для того чтобы не допустить потерю информации при порче диска, обычно используют резервное копирование файлов (*backup*). Простейшей формой *backup*'а является копирование всех файлов из одной директории в другую. Этот способ требует много времени и места на диске. Напишите программу, осуществляющую более интеллектуальный подход.

Программа должна брать из командной строки два параметра: исходную директорию и директорию назначения. Она должна рекурсивно сканировать исходную директорию, делать копии всех файлов, для которых ранее не делались копии или которые были изменены с момента последнего *backup*'а, размещая их в соответствующих местах директории назначения.

После копирования каждого файла должна вызываться команда сжатия *gzip*. Это уменьшит требуемый размер дисковой памяти; а файл будет переименован с добавлением расширения *.gz*. Все возникающие ошибки (нет исходной директории, файл закрыт для чтения и т.д.) должны корректно обрабатываться с выдачей соответствующего сообщения.

Задание 2

1. Напишите программу *runsim*, осуществляющую контроль количества одновременно работающих UNIX-приложений. Программа читает UNIX-команду со стандартного ввода и запускает ее на выполнение. Количество одновременно работающих команд не должно превышать *N*, где *N* – параметр командной строки при запуске *runsim*. При попытке запустить более чем *N* приложений выдайте сообщение об ошибке. Программа *runsim* должна прекращать свою работу при возникновении признака конца файла на стандартном вводе.

2. На мойке посуды в ресторане работают два человека. Один из них моет посуду, второй вытирает уже вымытую. Времена выполнения операций мытья и вытирания посуды меняются в зависимости от того, что моет. Стол для вымытой, но не вытертой посуды имеет ограниченные размеры.

Смоделируйте процесс работы персонала следующим образом: каждому работнику соответствует свой процесс. Времена выполнения операций содержатся в двух файлах. Каждый файл имеет формат записей:

<тип посуды> : <время операции>

Стол вмещает *N* предметов независимо от их наименования. Значение *N* задается как параметр среды *TABLE_LIMIT* перед стартом процессов. Грязная посуда, поступающая на мойку, описывается файлом с форматом записи:

<тип посуды> : <количество предметов>

Записи с одинаковым типом посуды могут встречаться неоднократно.

Организируйте передачу посуды от процесса к процессу:

- а) через файл, используя семафоры для синхронизации;
- б) через *pipe*;
- в) через сообщения;
- г) через разделяемую память;
- д) через *sockets*.

Примеры вопросов и задач для контрольных работ 3 (осенний) семестр

Часть 1.

1. Какая техническая база характерна для первого периода вычислительной техники (1945-1955 г.г.)?
 - полупроводники (транзисторы, диоды и т.д.)
 - интегральные микросхемы
 - электронные лампы
2. Что было прообразом современных ОС?
 - компиляторы с символических языков
 - библиотеки математических и служебных программ
 - системы пакетной обработки
3. К чему относится термин спулинг (spooling)?
 - к сбору заданий с одинаковым набором ресурсов в пакеты заданий
 - к организации реального ввода пакета заданий и вывода результатов на отдельных специализированных ЭВМ
 - к организации реального ввода пакета заданий и вывода результатов на том же компьютере, который производит вычисления
4. Планирование заданий стало возможным:
 - с появлением систем пакетной обработки
 - с появлением предварительной записи пакета заданий на магнитную ленту
 - с появлением предварительной записи пакета заданий на магнитный диск
5. Что из ниже перечисленного является мультипрограммными вычислительными системами?
 - система, в которой реализован спулинг (spooling)
 - система, в памяти которой одновременно находится несколько программ. Когда одна из программ ожидает завершения операции ввода-вывода, другая программа может выполняться
 - система, в памяти которой находится несколько программ, чье исполнение чередуется по прошествии определенного промежутка времени
6. Возможность интерактивного взаимодействия пользователя и программы возникает с появлением:
 - систем пакетной обработки
 - мультипрограммных вычислительных систем
 - систем разделения времени
7. Разделение персонала, связанного с разработкой и эксплуатацией ЭВМ, на разработчиков, специалистов по эксплуатации, операторов и программистов произошло:
 - в первый период развития вычислительной техники (1945-55 г.г.)
 - во второй период развития вычислительной техники (1955- начало 60-х г.г.)
 - в третий период развития вычислительной техники (начало 60-х - 1980 г.г.)
8. При доступе к файлу в сетевой ОС пользователь должен знать:
 - только имя файла
 - точное физическое расположение файла на диске
 - имя файла, компьютер, на котором находится файл, и сетевой способ доступа к информации в файле

9. При доступе к файлу в распределенной ОС пользователь должен знать:
- только имя файла
 - точное физическое расположение файла на диске
 - имя файла, компьютер, на котором находится файл, и сетевой способ доступа к информации в файле
10. Из каких состояний процесс может перейти в состояние *ожидание*?
- из состояния *рождение*
 - из состояния *готовность*
 - из состояния *исполнение*
11. Из каких состояний процесс может перейти в состояние *исполнение*?
- из состояния *ожидание*
 - из состояния *готовность*
 - из состояния *рождение*
12. Когда процесс, находящийся в состоянии *закончил исполнение*, может окончательно покинуть систему?
- по прошествии определенного интервала времени
 - только при перезагрузке операционной системы
 - после завершения процесса-родителя
13. Какие из перечисленных ниже компонентов входят в пользовательский контекст процесса?
- состояние, в котором находится процесс
 - программный счетчик процесса
 - информация об устройствах ввода-вывода, связанных с процессом
 - содержимое регистров процессора
 - код и данные, находящиеся в адресном пространстве процесса
14. Какие из перечисленных ниже компонентов входят в регистровый контекст процесса?
- состояние, в котором находится процесс
 - программный счетчик процесса
 - информация об устройствах ввода-вывода, связанных с процессом
 - содержимое регистров процессора
 - код и данные, находящиеся в адресном пространстве процесса
15. Какие из перечисленных ниже компонентов входят в системный контекст процесса?
- состояние, в котором находится процесс
 - программный счетчик процесса
 - информация об устройствах ввода-вывода, связанных с процессом
 - содержимое регистров процессора
 - код и данные, находящиеся в адресном пространстве процесса
16. При модернизации некоторой операционной системы, поддерживающей только три состояния процессов: *готовность*, *исполнение*, *ожидание*, принято решение ввести два новых системных вызова. Один из этих вызовов позволяет любому процессу приостановить жизнедеятельность любого другого процесса (кроме самого себя), до тех пор, пока какой-либо процесс не выполнит второй системный вызов. Сколько новых состояний процессов появится в системе?
- 1
 - 2
 - 3
17. При модернизации некоторой операционной системы, поддерживающей только три состояния процессов: *готовность*, *исполнение*, *ожидание*, решено ввести два новых системных вызова. Один из этих вызовов позволяет любому процессу приостановить жизнедеятельность любого другого процесса (кроме самого себя),

до тех пор, пока какой-либо процесс не выполнит второй системный вызов. Сколько новых операций над процессами появится в системе?

- 2
- 4
- 5

18. При модернизации некоторой операционной системы, поддерживающей только три состояния процессов: *готовность*, *исполнение*, *ожидание*, решено ввести два новых системных вызова. Один из этих вызовов позволяет любому процессу приостановить жизнедеятельность любого другого процесса (кроме самого себя), до тех пор, пока какой-либо процесс не выполнит второй системный вызов. Сколько новых переходов из состояния *исполнение* появится в системе?

- 0
- 2
- 4

19. Сколько операций над процессами существует в курсе «Операционные системы семейства UNIX»?

- 4
- 6
- 7

20. Какая операция над процессом в модели, принятой в данном курсе, не имеет парной?

- создание процесса
- запуск процесса
- изменение приоритета процесса

21. В некоторой операционной системе, похожей на UNIX, существует единственный способ порождения нового процесса, который будет являться дубликатом родительского процесса по регистровому и пользовательскому контекстам, с помощью системного вызова `fork()`. Для замены пользовательского контекста процесса и запуска исполняемого файла с именем `a.out` применяется системный вызов `exec("a.out")`. Неопытный программист написал следующую программу:

```
void main()
```

```
{ int i; for (i = 0; i < n; i++){ fork(); fork(); exec("a.out"); } while(1); }
```

где n - некоторая положительная константа. Сколько процессов будет запущено в операционной системе в результате ее выполнения, если программа `a.out` не запускает новых процессов?

- 2
- 4
- 2^n

22. В каких случаях производится невытесняющее кратковременное планирование процессов?

- когда процесс переводится из состояния *исполнение* в состояние *завершил исполнение*;
- когда процесс переводится из состояния *исполнение* в состояние *ожидание*;
- когда процесс переводится из состояния *ожидание* в состояние *готовность*.

23. На каких параметрах может основываться долгосрочное планирование процессов?

- на статических параметрах вычислительной системы;
- на динамических параметрах вычислительной системы;
- на статических параметрах процессов;
- на динамических параметрах процессов.

24. Какие из перечисленных алгоритмов допускают неограниченно долгое откладывание выборки одного из готовых процессов на исполнение?

- FCFS

- SJF
 - RR
 - многоуровневые очереди
25. Какие из перечисленных алгоритмов представляют собой частные случаи планирования с использованием приоритетов?
- FCFS
 - RR
 - SJF
 - гарантированное планирование
26. К какому из перечисленных алгоритмов стремится поведение алгоритма RR по мере увеличения кванта времени?
- SJF
 - FCFS
 - гарантированное планирование при одном процессе на каждого пользователя.
27. К какому из перечисленных алгоритмов теоретически стремится поведение алгоритма RR по мере уменьшения кванта времени:
- SJF
 - FCFS
 - гарантированное планирование при одном процессе на каждого пользователя
28. Пусть в вычислительную систему поступают пять процессов различной длительности по следующей схеме:

Номер процесса	Момент поступления в систему	Время исполнения
1	2	4
2	1	3
3	4	5
4	3	2
5	0	9

Чему равно среднее время ожидания процесса (waiting time) при использовании вытесняющего алгоритма SJF? При вычислениях считать, что процессы не совершают операций ввода-вывода, временем переключения контекста пренебречь.

- 11.3
 - 5.0
 - 8.4
29. Пусть в вычислительную систему поступают пять процессов различной длительности по следующей схеме:

Номер процесса	Момент поступления в систему	Время исполнения
1	2	4
2	1	3
3	4	5
4	3	2
5	0	9

Чему равно среднее время ожидания процесса (waiting time) при использовании невытесняющего алгоритма SJF? При вычислениях считать, что процессы не совершают операций ввода-вывода, временем переключения контекста пренебречь.

- 11.3
- 5.0
- 8.4

30. Пусть в вычислительную систему поступают пять процессов различной длительности с разными приоритетами по следующей схеме:

Номер процесса	Момент поступления в систему	Время исполнения	Приоритет
1	3	10	1
2	6	4	0
3	0	4	3
4	2	1	4
5	4	3	2

Чему равно среднее время между стартом процесса и его завершением (turnaround time) при использовании вытесняющего приоритетного планирования? При вычислениях считать, что процессы не совершают операций ввода-вывода, временем переключения контекста пренебречь. Наивысшим приоритетом является приоритет 0.

- 10.6
 - 13.4
 - 15.0
31. Какая категория средств связи получила наибольшее распространение в вычислительных системах?
- сигнальные
 - канальные
 - разделяемая память
32. Какой из вариантов адресации может использоваться для организации передачи информации через pipe?
- симметричная прямая адресация
 - асимметричная прямая адресация
 - непрямая адресация
33. Какое из перечисленных условий надежности связи не может быть выполнено со стопроцентной гарантией при выполнении остальных условий?
- не происходит потери информации
 - не происходит повреждения информации
 - не нарушается порядок данных в процессе обмена
34. Сколько процессов могут одновременно использовать одно и то же средство связи, пользуясь симметричной прямой адресацией?
- 2
 - произвольное количество
 - ответ зависит от того, является ли средство связи дуплексным или симплексным
35. Какие процессы могут обмениваться информацией через FIFO?
- только процесс, создавший FIFO, и его процесс-ребенок
 - только процессы, имеющие общего родителя, создавшего FIFO
 - произвольные процессы в системе
36. Какие процессы могут обмениваться информацией через pipe?
- только процесс, создавший pipe, и его непосредственный процесс-ребенок
 - только процессы, имеющие общего родителя, создавшего pipe
 - произвольные процессы в системе
37. В операционных системах, поддерживающих нити исполнения (threads) внутри одного процесса на уровне ядра системы, процесс находится в состоянии *готовность*, если:
- хотя бы одна нить процесса находится в состоянии *готовность*
 - хотя бы одна нить исполнения находится в состоянии *готовность*, и нет ни одной нити в состоянии *ожидание*

- хотя бы одна нить процесса находится в состоянии *готовность*, и нет ни одной нити в состоянии *исполнение*.
38. В операционных системах, поддерживающих нити исполнения (threads) внутри одного процесса на уровне ядра системы, наряду с блоками управления процессами (PCB) существуют структуры данных для управления нитями - TCB (Thread Control Block). Укажите, какие данные из перечисленных ниже хранятся, по вашему мнению, в TCB:
- данные о файлах, используемых процессом
 - указатель стека
 - идентификатор пользователя, инициировавшего работу процесса
39. В операционных системах, поддерживающих нити исполнения (threads) внутри одного процесса на уровне ядра системы, наряду с блоками управления процессами (PCB) существуют структуры данных для управления нитями - TCB (Thread Control Block). Укажите, какие данные из перечисленных ниже хранятся, по вашему мнению, в TCB:
- содержимое регистров процессора
 - данные, описывающие расположение адресного пространства процессора
 - приоритет нити исполнения
40. Рассмотрим две активности, P и Q:
- | | |
|---------|---------|
| P: | Q: |
| $y=x+2$ | $z=x-3$ |
| $f=y-4$ | $f=z+1$ |
- Набор из этих двух активностей является:
- детерминированным
 - недетерминированным
 - детерминированность зависит от значения x
41. Рассмотрим две активности, P и Q:
- | | |
|---------|---------|
| P: | Q: |
| $y=x+1$ | $z=x-3$ |
| $f=y-4$ | $f=z+1$ |
- Набор из этих двух активностей является:
- детерминированным
 - недетерминированным
 - детерминированность зависит от значения x
42. Если для некоторого набора активностей условия Бернштейна не выполняются, то набор активностей является:
- детерминированным
 - недетерминированным
 - может быть как недетерминированным, так и детерминированным
43. Прием взаимоисключения применяется:
- для того чтобы у процесса не было критического участка
 - для устранения условия гонки
 - для того чтобы процессы не использовали одни и те же ресурсы.
44. Термин race condition (условие гонки) относится:
- к набору процессов, совместно использующих какой-либо ресурс
 - к набору процессов, демонстрирующих недетерминированное поведение
 - к набору процессов, для каждого из которых важно завершиться как можно быстрее
45. Термин «критическая секция» относится:
- к участку процесса с наибольшим объемом вычислительной работы
 - к участку процесса, в котором процесс совместно с другими процессами использует разделяемые переменные

- к участку процесса, выполнение которого совместно с другими процессами может привести к неоднозначным результатам
46. Какие из условий для организации корректного взаимодействия двух процессов с помощью программного алгоритма выполнены для алгоритма «переменная-замок»:
- условие взаимоисключения
 - условие прогресса
 - условие ограниченного ожидания
47. Какие из условий для организации корректного взаимодействия двух процессов с помощью программного алгоритма выполнены для алгоритма «строгое чередование»:
- условие взаимоисключения
 - условие прогресса
 - условие ограниченного ожидания
48. Какие из условий для организации корректного взаимодействия двух процессов с помощью программного алгоритма выполнены для алгоритма «флаги готовности»:
- условие взаимоисключения
 - условие прогресса
 - условие ограниченного ожидания
49. В функциях-методах мониторов Хора обычно реализовываются:
- только прологи и эпилоги критических участков
 - критические участки взаимодействующих процессов
 - только различные операции над внутренними переменными монитора (как операции над внутренними переменными класса в ООП)
50. Условные переменные в мониторах Хора обычно используются:
- для обеспечения взаимоисключения в критических участках кооперативных процессов
 - для обеспечения взаимосинхронизации кооперативных процессов
 - для передачи данных между кооперативными процессами
51. Какие из перечисленных механизмов синхронизации обычно реализуются в вычислительной системе с помощью специальных системных вызовов?
- семафоры Дейкстры
 - мониторы Хора
 - очереди сообщений
52. Отладка программ, содержащих очень большое количество семафоров, затруднена, так как:
- требует специального программного обеспечения
 - ошибочные ситуации трудно воспроизводимы
 - для хорошего программиста никаких затруднений не возникает
53. Для чего нужен синхронизирующий процесс при реализации семафоров через очереди сообщений:
- для удобства реализации
 - для обеспечения взаимной синхронизации кооперативных процессов
 - для обеспечения атомарности операций P и V
54. Рассмотрим механизм синхронизации, называемый бинарными семафорами. Бинарный семафор — это семафор, который может принимать всего два значения: 0 и 1. Операция P для этого семафора выглядит так же, как и для семафора Дейкстры, а операция V заключается в простом присваивании семафору значения 1. Бинарные семафоры:
- обладают меньшими возможностями, чем семафоры Дейкстры
 - обладают большими возможностями, чем семафоры Дейкстры
 - эквивалентны семафорам Дейкстры
55. На каком уровне иерархии памяти находится программа в процессе выполнения?

- на магнитном диске
 - в оперативной памяти
 - разные компоненты программы могут находиться на различных уровнях
56. Чем обусловлена эффективность иерархической схемы памяти?
- скоростью обмена с оперативной памятью
 - принципом локальности
 - количеством уровней в иерархии.
57. На каком этапе осуществляется связывание логического и физического адресных пространств процесса в современных ОС?
- на этапе выполнения
 - на этапе загрузки
 - на этапе компиляции
58. Внутренняя фрагментация - это:
- потеря части памяти, выделенной процессу, но не используемой им
 - разбиение адресного пространства процесса на фрагменты
 - потери части памяти в схеме с динамическими (переменными) разделами
59. Возможность организации в больших программах структур с перекрытиями (overlays) обусловлена:
- наличием в программе большого количества независимых процедур
 - разбиением памяти на несколько фиксированных разделов
 - принципом локальности
60. Что понимается под термином «внешняя фрагментация»?
- потеря части памяти, не выделенной ни одному процессу
 - потеря части памяти в схеме с фиксированными разделами
 - наличие фрагментов памяти, внешних по отношению к процессу
61. Какая из схем управления памятью подвержена внешней фрагментации?
- схема с фиксированными разделами
 - схема с динамическими разделами
 - страничная организация
62. Какая из схем управления памятью подвержена внутренней фрагментации?
- схема с фиксированными разделами
 - сегментная организация
 - страничная организация
63. Таблица страниц процесса - это:
- структура, используемая для отображения логического адресного пространства в физическое при страничной организации памяти
 - структура, организованная для учета свободных и занятых страничных блоков
 - структура, организованная для контроля доступа к страницам процесса
64. В чем состоит преимущество схемы виртуальной памяти по сравнению с организацией структур с перекрытием?
- возможность выполнения программ большего размера
 - возможность выполнения программ, размер которых превышает размер оперативной памяти
 - экономия времени программиста при размещении в памяти больших программ
65. Чем запись в таблице страниц в схеме страничной виртуальной памяти отличается от соответствующей записи в случае простой страничной организации?
- наличием номера страничного кадра
 - наличием бита присутствия
 - наличием атрибутов защиты страницы

66. Какая из рассмотренных ниже схем управления памятью пригодна для организации виртуальной памяти?
- страничная
 - сегментная
 - схема с фиксированными разделами
 - схема с динамическими разделами
67. Разбиение логического адресного пространства процесса на страницы в страничной схеме управления памятью осуществляется:
- программистом, поскольку он может сделать это самым оптимальным образом
 - компилятором, который может выполнить размещение различных компонентов программы на разных страницах
 - менеджером памяти ОС и блоком управления памятью незаметно для программиста
68. Вычислите номер страницы памяти и смещение для логического адреса 32768, если размер страницы равен 4Кб. Страницы нумеруются, начиная с 0.
- 8 и 0
 - 5 и 4096
 - 7 и 0
69. Сколько записей содержится в одноуровневой таблице страниц в системе с 32-разрядной архитектурой и размером страницы 4Кб?
- 2^{32}
 - 2^{20}
 - 2^{12}
70. Предположим, что для организации двухуровневой таблицы страниц 32-разрядный логический адрес разбивается на три поля: a,b,c. Третье поле — c — это смещение внутри страницы. От чего зависит количество страниц в логическом адресном пространстве процесса?
- от размера всех трех полей
 - от размера поля c
 - от размера поля a
71. В вычислительной системе со страничной организацией памяти и 32-х битовым адресом размер страницы составляет 8 Мбайт. Для некоторого процесса таблица страниц в этой системе имеет вид
- | | |
|---|------------|
| 1 | 0x00000000 |
| 2 | 0x02000000 |
| 5 | 0x06000000 |
| 6 | 0x10000000 |
- Какому физическому адресу соответствует логический адрес 0x00827432:
- 0x27432
 - 0x02027432
 - 0x10027432
72. В диком каннибальском племени вокруг котла с пищей спят дикари и повар. Изначально в котле находится N порций мяса. Дикари по очереди просыпаются, берут из котла порцию мяса, съедают его и засыпают снова. Дикарь, не обнаруживший мяса в котле, будит повара. Повар находит добычу и снова готовит N порций, не подпуская никого к котлу во время приготовления, после чего тоже засыпает. Используя семафоры Дейкстры и, при необходимости, разделяемые переменные, постройте корректную модель происходящего, описав поведение каждого из дикарей и повара с помощью отдельных процессов.

73. В диком каннибальском племени вокруг котла с пищей спят дикари и повар. Изначально в котле находится N порций мяса. Дикари по очереди просыпаются, берут из котла порцию мяса, съедают его и засыпают снова. Дикарь, не обнаруживший мяса в котле, будит повара. Повар находит добычу и снова готовит N порций, не подпуская никого к котлу во время приготовления, после чего тоже засыпает. Используя мониторы Хора, постройте корректную модель происходящего, описав поведение каждого из дикарей и повара с помощью отдельных процессов.
74. В диком каннибальском племени вокруг котла с пищей спят дикари и повар. Изначально в котле находится N порций мяса. Дикари по очереди просыпаются, берут из котла порцию мяса, съедают его и засыпают снова. Дикарь, не обнаруживший мяса в котле, будит повара. Повар находит добычу и снова готовит N порций, не подпуская никого к котлу во время приготовления, после чего тоже засыпает. Используя классические очереди сообщений, постройте корректную модель происходящего, описав поведение каждого из дикарей и повара с помощью отдельных процессов.

Часть 2.

75. Известно, что время доступа к оперативной памяти 100 нс, а время доступа к ассоциативной памяти – 10 нс. Частота попаданий в ассоциативную память при обращении к данным (hit ratio) составляет 90%. Чему равно среднее время обращения за данным к памяти для одноуровневой таблицы страниц?
- 19 нс
 - 120 нс
 - 168 нс
76. Для оповещения вычислительной системы об отсутствии нужной страницы в памяти используется:
- механизм системных вызовов
 - механизм аппаратных прерываний
 - механизм исключительных ситуаций
77. Какую стратегию управления памятью может реализовать алгоритм выталкивания страниц LRU?
- стратегию размещения страницы в памяти при наличии списка свободных кадров
 - стратегию упреждающей выборки, когда кроме страницы, вызвавшей исключительную ситуацию, в память также загружается несколько страниц, окружающих ее
 - стратегию замещения
78. Какой результат может иметь анализ бита модификации, входящего в состав атрибутов страницы?
- уменьшение времени обработки page fault'a ввиду того, что копия страницы уже имеется на диске
 - необходимость коррекции записи о странице в таблице страниц, поскольку содержимое страницы изменено
 - блокировку страницы в памяти для того, чтобы сохранить изменения содержимого страницы в неприкосновенности
79. Какие действия обычно выполняет операционная система в том случае, если процесс обратился к отсутствующей странице, а свободных кадров в наличии нет?
- приостанавливает этот процесс до тех пор, пока в системе не появятся свободные кадры
 - приостанавливает процесс, находит некоторый занятый кадр основной памяти, перемещает в случае надобности его содержимое во внешнюю память, переписывает в этот страничный кадр содержимое логической

страницы из внешней памяти, после чего возобновляет выполнение процесса

- выгружает процесс на диск

80. Для некоторого процесса, запущенного в вычислительной системе со страничной организацией памяти с использованием LRU алгоритма замещения страниц, выделение процессу 4 кадров памяти приводит к 11 page fault'ам, а выделение 6 кадров памяти – к 9 page fault'ам (вначале все кадры свободны). Какой (какие) вариант(ы) количества page fault'ов для того же процесса и того же количества кадров может быть получен при использовании OPT алгоритма замещения страниц?

- 12 и 8
- 8 и 7
- 7 и 8
- 9 и 6

81. Для некоторого процесса известна следующая строка запросов страниц памяти 1, 5, 2, 3, 2, 1, 4, 5, 1, 2, 3, 4, 1, 5, 2, 1, 3, 2, 4. Сколько ситуаций отказа страницы (*page fault*) возникнет для данного процесса при использовании алгоритма замещения страниц FIFO (First Input First Output), если процессу выделено 3 кадра памяти?

- 15
- 12
- 10

82. Для некоторого процесса известна следующая строка запросов страниц памяти 1, 5, 2, 3, 2, 1, 4, 5, 1, 2, 3, 4, 1, 5, 2, 1, 3, 2, 4. Сколько ситуаций отказа страницы (*page fault*) возникнет для данного процесса при использовании алгоритма замещения страниц LRU (the Least Recently Used), если процессу выделено 3 кадра памяти?

- 15
- 12
- 10

83. Для некоторого процесса известна следующая строка запросов страниц памяти 1, 5, 2, 3, 2, 1, 4, 5, 1, 2, 3, 4, 1, 5, 2, 1, 3, 2, 4. Сколько ситуаций отказа страницы (*page fault*) возникнет для данного процесса при использовании алгоритма замещения страниц OPT (optimal), если процессу выделено 3 кадра памяти?

- 15
- 12
- 10

84. Файловая система включается в состав ОС для того, чтобы:

- эффективно использовать дисковое пространство
- обеспечить пользователя удобным интерфейсом для работы с внешней памятью
- повысить производительность системы ввода-вывода

85. Известно, что в большинстве ОС файл представляет собой неструктурированную последовательность байтов и хранится на диске. Какой способ доступа обычно применяется к таким файлам?

- последовательный
- прямой
- индексно-последовательный

86. Для чего по окончании работы с файлом принято выполнять операцию закрытия (close) файла?

- чтобы освободить место во внутренних таблицах файловой системы
- чтобы перевести указатель текущей позиции в начало файла
- чтобы разрешить доступ к файлу другим процессам

87. Входит ли имя каталога, в котором находится файл, в полное имя файла на диске?

- не входит
 - входит
 - это зависит от того, является данный каталог рабочим
88. Схема выделения дискового пространства связным списком блоков не нашла широкого применения, так как:
- неэффективно использует дисковое пространство
 - требует большого количества обращений к диску при работе с файлами
 - страдает от внутренней фрагментации
89. Основным преимуществом использования таблицы отображения фалов (FAT) по сравнению с классической схемой выделения связным списком является:
- сокращение количества обращений к диску
 - повышенная надежность
 - более экономичное использование дискового пространства
90. Большинство файловых систем, поддерживаемых ОС Unix для выделения дискового пространства, использует схему:
- с индексными узлами
 - связного списка блоков
 - выделения непрерывной последовательности блоков
91. Предположим, что один из файлов в ОС Unix жестко связан с двумя различными каталогами, принадлежащими различным пользователям. Что произойдет, если один из пользователей удалит файл?
- файл автоматически удалится из каталога второго пользователя
 - содержание каталога второго пользователя не изменится
 - система отменит операцию удаления файла
92. Какие из перечисленных ситуаций возникают синхронно с работой процессора?
- прерывания
 - исключительные ситуации
 - программные прерывания
93. Какие из перечисленных ситуаций возникают предсказуемо?
- прерывания
 - исключительные ситуации
 - программные прерывания
94. Какие из перечисленных ситуаций обнаруживаются процессором между выполнением команд?
- прерывания
 - исключительные ситуации
 - программные прерывания
95. Какие из параметров запроса к жесткому диску обычно учитываются при планировании последовательности запросов?
- вид операции
 - номер сектора
 - номер цилиндра
 - номер дорожки
96. Какие из вариантов реализации системного вызова read могут прочитать меньше байт, чем запросил процесс?
- асинхронный
 - блокирующийся
 - неблокирующийся
97. Какие из перечисленных функций базовой подсистемы ввода-вывода могут быть делегированы драйверам?
- поддержка блокирующихся, неблокирующихся и асинхронных системных вызовов

- обработка ошибок и прерываний, возникающих при операциях ввода-вывода
 - буферизация и кэширование входных и выходных данных
 - планирование последовательности запросов на выполнение операций ввода-вывода
98. Пусть у нас имеется диск с 80 цилиндрами (от 0 до 79). Время перемещения головки между соседними цилиндрами составляет 1мс. Время же перевода головки с 79-го на 0-й цилиндр составляет всего 10 мс. В текущий момент времени головка находится на 45-м цилиндре и движется в сторону увеличения номеров цилиндров. Сколько времени будет обрабатываться следующая последовательность запросов на чтение цилиндров: 10, 6, 15, 71, 1, 62, для алгоритма SSTF (временами чтения цилиндров и смены направления движения пренебречь)?
- 121 мс
 - 96 мс
 - 59 мс
99. Пусть у нас имеется диск с 80 цилиндрами (от 0 до 79). Время перемещения головки между соседними цилиндрами составляет 1мс. Время же перевода головки с 79-го на 0-й цилиндр составляет всего 10 мс. В текущий момент времени головка находится на 45-ом цилиндре и движется в сторону увеличения номеров цилиндров. Сколько времени будет обрабатываться следующая последовательность запросов на чтение цилиндров: 10, 6, 15, 71, 1, 62, для алгоритма C-SCAN (временами чтения цилиндров и смены направления движения пренебречь)?
- 121 мс
 - 96 мс
 - 59 мс
100. Пусть у нас имеется диск с 80 цилиндрами (от 0 до 79). Время перемещения головки между соседними цилиндрами составляет 2 мс. В текущий момент времени головка находится на 23-м цилиндре и движется в сторону увеличения номеров цилиндров. Сколько времени будет обрабатываться следующая последовательность запросов на чтение цилиндров: 11, 22, 10, 73, 1, 12, алгоритма SCAN (временами чтения цилиндров и смены направления движения головок пренебречь)?
- 362 мс
 - 268 мс
 - 188 мс
101. Какие операционные системы позволяют взаимодействовать удаленным процессам и имеют сходное строение с автономными вычислительными системами?
- сетевые операционные системы
 - распределенные операционные системы
 - операционные системы, поддерживающие работу многопроцессорных вычислительных систем
102. Сколько удаленных адресов может иметь сетевой компьютер?
- только один
 - не более двух
 - потенциально произвольное количество
103. Пусть в некоторой сетевой операционной системе существует три различных протокола транспортного уровня, использующих собственные адресные пространства портов. Сколько типов сокетов существует в такой системе?
- один
 - три
 - зависит от реализации

104. Пусть у нас есть локальная вычислительная сеть, достаточно долгое время работающая с неизменной топологией и без сбоев. Какие алгоритмы маршрутизации гарантируют доставку пакетов данных от отправителя к получателю по кратчайшему пути?
- алгоритмы лавинной маршрутизации
 - алгоритмы состояния связей
 - маршрутизация от источника данных
105. Пусть у нас есть локальная вычислительная сеть, достаточно долгое время работающая с неизменной топологией и без сбоев. Какие алгоритмы маршрутизации гарантируют доставку пакетов данных по кратчайшему пути?
- алгоритмы фиксированной маршрутизации
 - векторно-дистанционные алгоритмы с метрикой количества переходов между компонентами сети
 - алгоритмы случайной маршрутизации
106. Какие категории средств связи используются при взаимодействии удаленных процессов?
- сигнальные
 - канальные
 - разделяемая память
107. Какой уровень эталонной модели OSI/ISO отвечает за создание контрольных точек при общении удаленных процессов?
- сетевой уровень
 - транспортный уровень
 - уровень сеанса
108. Какой уровень эталонной модели OSI/ISO отвечает за доставку информации от компьютера-отправителя к компьютеру-получателю?
- сетевой уровень
 - транспортный уровень
 - уровень сеанса
109. Какой уровень эталонной модели OSI/ISO отвечает за доставку информации от процесса-отправителя процессу-получателю?
- сетевой уровень
 - транспортный уровень
 - уровень приложений

4. Критерии оценивания

За каждый контрольный вопрос из контрольного задания студент получает от 0 до максимального балла в зависимости от полноты представленного ответа (решения). Критерии проставления баллов утверждаются на заседании учебно-методической комиссии кафедры. Процент суммарно набранных баллов от максимально возможного количества определяет оценку за теоретические знания по каждому контрольному заданию:

Оценка	Набранные баллы
отлично (10)	более 88%
отлично (9)	от 78% до 88% включительно
отлично (8)	от 68% до 78% включительно
хорошо (7)	от 58% до 68% включительно
хорошо (6)	от 48% до 58% включительно
хорошо (5)	от 38% до 48% включительно
удовлетворительно (4)	от 28% до 38% включительно

удовлетворительно (3)	от 18% до 28% включительно
неудовлетворительно (2)	от 08% до 18% включительно
неудовлетворительно (1)	не более 08%

Каждая лабораторная работа и задача из задания при полном правильном решении оценивается в определенное количество баллов от 5 до 25. Баллы за полностью правильное решение определяются преподавателями и утверждаются на заседании учебно-методической комиссии кафедры. Процент суммарно набранных баллов от максимально возможного количества определяет оценку за практические знания студента по вышеприведенной таблице.

Итоговая оценка за дифференцированный зачет выставляется на основании оценок, полученных за контрольные работы по теории (далее x и y), и оценки за практические знания, (далее z) по следующей формуле $(x+y)*z/(z+(x+y)/2)$, с округлением результата до целого сверху.

5. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности

Промежуточная аттестация по дисциплине «Операционные системы» в 3 (осеннем) семестре осуществляется в форме дифференцированного зачета. Дифференцированный зачет выставляется по результатам работы студента в течение семестра на основе оценок за 15 лабораторных работ, 2 задания, 2 контрольные работы.

Время проведения каждой контрольной работы составляет 2-4 академических часа.

Во время проведения контрольных работ обучающиеся могут пользоваться программой дисциплины и любыми материалами.