**Программа учебной дисциплины «Операционные системы»**

Утверждена

Академическим советом ООП

Протокол № от «\_\_»\_\_\_\_\_20\_\_ г.

|  |  |
| --- | --- |
| Автор | Карпов В.Е., к.ф.-м.н., доцент, carpson@mail.ru |
| Число кредитов |  |
| Контактная работа (час.) | 56 |
| Самостоятельная работа (час.) | 96 |
| Курс | 2 |
| Формат изучения дисциплины | Без использования online курса |

1. **ЦЕЛЬ, РЕЗУЛЬТАТЫ ОСВОЕНИЯ ДИСЦИПЛИНЫ И ПРЕРЕКВИЗИТЫ**

Целями освоения дисциплины «Операционные системы» являются освоение студентами знаний в области построения и функционирования современных операционных систем, осмысленное применение полученных знаний при изучении других дисциплин.

В результате освоения дисциплины студент должен:

**знать:**

* историю эволюции вычислительных систем, основные функции, выполняемые современными операционными системами, принципы их внутреннего построения;
* концепцию процессов в операционных системах;
* основные алгоритмы планирования процессов;
* логические основы взаимодействия процессов;
* концепцию нитей исполнения и их отличие от обычных процессов;
* программные алгоритмы организации взаимодействия процессов и предъявляемые к ним требования;
* основные механизмы синхронизации в операционных системах;
* организацию управления оперативной памятью использующиеся при этом алгоритмы;
* основные принципы управления файловыми системами;
* организацию управления устройствами ввода-вывода на уровне как технического, так и программного обеспечения, основные функции подсистемы ввода-вывода;
* принципы сетевого взаимодействия вычислительных систем и построения работы сетевых частей операционных систем;
* основные проблемы безопасности операционных систем и подходы к их решению.

**уметь:**

* пользоваться командами командного интерпретатора операционной системы Linux;
* порождать новые процессы, запускать новые программы и правильно завершать их функционирование;
* порождать новые нити исполнения и правильно завершать их функционирование;
* организовывать взаимодействие процессов через потоковые средства связи, разделяемую память и очереди сообщений;
* использовать семафоры и сигналы для синхронизации работы процессов и нитей исполнения;
* использовать системные вызовы для работы с файловой системой;
* разрабатывать программы для сетевого взаимодействия.

**владеть:**

* навыками использования команд командного интерпретатора в операционной системе Linux;
* навыками написания и отладки программ, порождающих несколько процессов или нитей исполнения;
* навыками написания и отладки программ, использующих системные вызовы для взаимодействия локальных процессов;
* навыками написания и отладки программ, использующих системные вызовы для работы с файловыми системами и устройствами ввода-вывода;
* навыками написания и отладки сетевых приложений.

Изучение дисциплины «Операционные системы» базируется на следующих дисциплинах:

- «Алгоритмы и структуры данных. Часть 1»;

- «Архитектура вычислительных систем».

Для освоения учебной дисциплины студенты должны владеть следующими знаниями и компетенциями:

* знать основные конструкции языков программирования C/C++;
* знать основные структуры данных, применяемые в языках программирования, и алгоритмы для работы с ними;
* обладать навыками написания программ на языках программирования C/C++ и их отладки.

Основные положения дисциплины должны быть использованы в дальнейшем при изучении следующих дисциплин:

1. Распределённые вычисления
2. НИС «Облачные технологии»;
3. НИС ПИ «Технологии программирования»;
4. **СОДЕРЖАНИЕ УЧЕБНОЙ ДИСЦИПЛИНЫ**

**Тема 1**. **Введение в дисциплину.**

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.

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.

**Тема 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.

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*().

**Тема 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. Потоки ввода‑вывода и сообщения. Буферизация данных. Надежность обмена информацией. Нити исполнения и их отличие от процессов. Interleaving, race condition и взаимоисключения. Условия Бернстайна. Понятие критической секции процесса. Программные алгоритмы организации взаимодействия процессов и предъявляемые к ним требования. Семафоры, мониторы Хора и сообщения.

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.

**Тема 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.

**Тема 5. 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.

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 *s*5*fs*. 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*.

**Тема 6. 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. 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 *SIGUSR*1 и *SIGUSR*2. Использование сигналов для синхронизации процессов. Понятие о надежности сигналов.

**Тема 7. 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.

**Тема 8. 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.

1. **ОЦЕНИВАНИЕ**

Для контроля знаний студентов используется текущий контроль (контрольная работа и домашнее задание), оценка за аудиторную работу и промежуточная аттестация (письменный экзамен). Контрольная работа проводится в 3-м модуле учебного года, домашнее задание дается в 4-м модуле учебного года.

Контрольная работа и письменный экзамен длительностью 80 минут состоят из 4-6 контрольных вопросов по теории с различными баллами от 2 до 18 за полностью правильный ответ для каждого вопроса, в зависимости от сложности. Перечень контрольных вопросов для контрольной работы, письменного экзамена и баллы за полностью правильный ответ утверждаются лектором.

За каждый контрольный вопрос из контрольной работы и письменного экзамена 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 контрольную работу и за письменный экзамен:

|  |  |
| --- | --- |
| 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 |
| удовлетворительно (5) | от 38% до 48% включительно |
| удовлетворительно (4) | от 28% до 38% включительно |
| неудовлетворительно (3) | от 18% до 28% включительно |
| неудовлетворительно (2) | от 08% до 18% включительно |
| неудовлетворительно (1) | не более 08% |

Домашнее задание содержит 2 практических задачи, с различными баллами от 5 до 25 за полностью правильное решение, в зависимости от сложности.

За каждую задачу из домашнего задания студент получает от 0 до максимального балла в зависимости от полноты представленного решения. Процент суммарно набранных баллов от максимально возможного количества определяет оценку за домашнее задание по вышеприведенной таблице.

Преподаватель оценивает работу студентов на лабораторных занятиях. Каждая лабораторная работа при полном правильном решении оценивается в определенное количество баллов от 5 до 25. Баллы за полностью правильное решение определяются преподавателями и утверждаются лектором. Процент суммарно набранных баллов от максимально возможного количества определяет оценку за практические знания студента по вышеприведенной таблице. Накопленная оценка по 10-ти балльной шкале за работу на практических занятиях определяется перед итоговым контролем - *Оаудиторная*.

Накопленная оценка за текущий контроль учитывает результаты студента по текущему контролю следующим образом:

О*накопленная*= 0,41\* *Отекущий* + 0,59\* *Оаудиторная* ,

где *Отекущий* рассчитывается как взвешенная сумма всех форм текущего контроля

*Отекущий* = *0,8·Ок/р + 0,2·Одз* ;

Округление проводится арифметическим способом.

Результирующая оценка за дисциплину рассчитывается следующим образом:

*Орезульт = 0,75\* Онакопл + 0,25 \*·Оэкз*

Округление проводится арифметическим способом.

1. **ПРИМЕРЫ ОЦЕНОЧНЫХ СРЕДСТВ**

# **Оценочные средства для текущего контроля студента**

Примерные вопросы для контрольной работы

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 исполняться
   * 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
* 2n

1. 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.*
2. 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.
3. Какие из перечисленных алгоритмов допускают неограниченно долгое откладывание выборки одного из готовых процессов на исполнение?
   * FCFS
   * SJF
   * RR
   * многоуровневые очереди
4. Какие из перечисленных алгоритмов представляют собой частные случаи планирования с использованием приоритетов?
   * FCFS
   * RR
   * SJF
   * гарантированное планирование
5. К какому из перечисленных алгоритмов стремится поведение алгоритма RR по мере увеличения кванта времени?
   * SJF
   * FCFS
   * гарантированное планирование при одном процессе на каждого пользователя.
6. 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
7. 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

1. 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

1. 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 (turnaroud 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

1. Какая категория средств связи получила наибольшее распространение в вычислительных системах?
   * сигнальные
   * канальные
   * разделяемая память
2. Какой из вариантов адресации может использоваться для организации передачи информации через pipe?
   * симметричная прямая адресация
   * асимметричная прямая адресация
   * непрямая адресация
3. 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
4. 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
5. 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
6. 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
7. 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.*
8. 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
9. 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
10. 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

1. 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

1. Если для некоторого набора активностей условия Бернстайна не выполняются, то набор активностей является:
   * детерминированным
   * недетерминированным
   * может быть как недетерминированным, так и детерминированным
2. 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.
3. 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
4. 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
5. 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
6. 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
7. 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
8. 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)
9. 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
10. Какие из перечисленных механизмов синхронизации обычно реализуются в вычислительной системе с помощью специальных системных вызовов?
    * семафоры Дейкстры
    * мониторы Хора
    * очереди сообщений
11. Отладка программ, содержащих очень большое количество семафоров, затруднена, так как:
    * требует специального программного обеспечения
    * ошибочные ситуации трудновоспроизводимы
    * для хорошего программиста никаких затруднений не возникает
12. 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
13. 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
14. На каком уровне иерархии памяти находится программа в процессе выполнения?
    * на магнитном диске
    * в оперативной памяти
    * разные компоненты программы могут находиться на различных уровнях
15. Чем обусловлена эффективность иерархической схемы памяти?
    * скоростью обмена с оперативной памятью
    * принципом локальности
    * количеством уровней в иерархии.
16. На каком этапе осуществляется связывание логического и физического адресных пространств процесса в современных ОС?
    * на этапе выполнения
    * на этапе загрузки
    * на этапе компиляции
17. 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
18. 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
19. 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
20. Какая из схем управления памятью подвержена внешней фрагментации?
    * схема с фиксированными разделами
    * схема с динамическими разделами
    * страничная организация
21. Какая из схем управления памятью подвержена внутренней фрагментации?
    * схема с фиксированными разделами
    * сегментная организация
    * страничная организация
22. 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
23. 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
24. 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
25. Какая из рассмотренных ниже схем управления памятью пригодна для организации виртуальной памяти?
    * страничная
    * сегментная
    * схема с фиксированными разделами
    * схема с динамическими разделами
26. 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
27. 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
28. Сколько записей содержится в одноуровневой таблице страниц в системе с 32-разрядной архитектурой и размером страницы 4Кб?
    * 232
    * 220
    * 212
29. Предположим, что для организации двухуровневой таблицы страниц 32-разрядный логический адрес разбивается на три поля: a,b,c. Третье поле — с — 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 а
30. В вычислительной системе со страничной организацией памяти и 32-х битовым адресом размер страницы составляет 8 Mбайт. Для некоторого процесса таблица страниц в этой системе имеет вид

|  |  |
| --- | --- |
| 1 | 0x00000000 |
| 2 | 0x02000000 |
| 5 | 0x06000000 |
| 6 | 0x10000000 |

Какому физическому адресу соответствует логический адрес 0х00827432:

* + 0x27432
  + 0x02027432
  + 0x10027432

1. 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.
2. 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.
3. 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.

Примерные задачи для домашнего задания

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 моется. 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.

Организуйте 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.*

**Оценочные средства для промежуточной аттестации**

Примерный перечень вопросов к экзамену для самопроверки студентов.

1. Понятие о вычислительном комплексе. Системное программное обеспечение и операционные системы.
2. Краткая история эволюции вычислительных систем. Взаимное влияние software и hardware.
3. Автономные, сетевые и распределенные операционные системы.
4. Классификация автономных операционных систем по их назначению и структуре.
5. Понятие процесса. Процесс и программа.
6. Состояния процесса.
7. Управляющий блок процесса и его контекст.
8. Операции над процессами.
9. Переключение контекста.
10. Уровни планирования процессов.
11. Критерии планирования и требования к алгоритмам планирования.
12. Параметры планирования.
13. Вытесняющее и невытесняющее планирование.
14. Алгоритмы планирования: FCFS, RR, SJF,
15. Гарантированное планирование, приоритетное планирование,
16. Многоуровневые очереди, многоуровневые очереди с обратной связью.
17. Взаимодействующие и независимые процессы.
18. Категории средств связи.
19. Установление и завершение связи.
20. Прямая и косвенная адресация.
21. Информационная валентность процессов и средств коммуникации.
22. Симплексная, дуплексная и полудуплексная связь.
23. Потоки ввода‑вывода и сообщения.
24. Буферизация данных.
25. Надежность обмена информацией.
26. Нити исполнения и их отличие от процессов.
27. Interleaving, race condition и взаимоисключения.
28. Условия Бернстайна.
29. Понятие критической секции процесса.
30. Программные алгоритмы организации взаимодействия процессов и предъявляемые к ним требования.
31. Аппаратная поддержка взаимоисключений.
32. Семафоры, мониторы Хора и сообщения.
33. Доказательство эквивалентности мониторов, семафоров и сообщений.
34. Связывание адресов.
35. Простейшие схемы управления памятью: схема с фиксированными разделами, своппинг, схема с переменными разделами.
36. Проблема размещения больших программ.
37. Понятие виртуальной памяти.
38. Страничная память.
39. Сегментная и сегментно-страничная организации памяти.
40. Таблица страниц.
41. Ассоциативная память.
42. Исключительные ситуации при работе с памятью.
43. Стратегии управления страничной памятью: выборки, размещения и замещения страниц.
44. Алгоритмы замещения страниц: FIFO, OPT, LRU и другие.
45. Thrashing.
46. Свойство локальности.
47. Модель рабочего множества.
48. Аппаратно-независимая модель памяти процесса.
49. Структура контроллера устройства.
50. Опрос устройств и прерывания.
51. Исключительные ситуации и системные вызовы.
52. Прямой доступ к памяти (Direct Memory Access – DMA).
53. Структура системы ввода-вывода.
54. Систематизация внешних устройств и интерфейс между базовой подсистемой ввода-вывода и драйверами.
55. Функции базовой подсистемы ввода-вывода.
56. Блокирующиеся, не блокирующиеся и асинхронные системные вызовы.
57. Буферизация и кэширование.
58. Spooling и захват устройств.
59. Обработка прерываний и ошибок.
60. Планирование запросов к устройству.
61. Алгоритмы планирования запросов к жесткому диску: FCFS, SSTF, SCAN, C-SCAN, LOOK, C-LOOK.
62. Взаимодействие удаленных процессов как основа работы вычислительных сетей.
63. Удаленная адресация и разрешение адресов.
64. Понятие о DNS.
65. Локальная адресация. Понятие порта.
66. Полные адреса. Понятие сокета (socket).
67. Фиксированная, виртуальная и динамическая маршрутизация.
68. Многоуровневая модель построения сетевых вычислительных систем.
69. Семейства и стеки протоколов.
70. Эталонная модель OSI/ISO.
71. Классификация угроз.
72. Формализация подхода к обеспечению информационной безопасности. Классы безопасности. Политика безопасности.
73. Шифрование с симметричными и ассиметричными ключами.
74. Правило Кирхгофа. Алгоритм RSA.
75. Идентификация и аутентификация.
76. Пароли, уязвимость паролей. Авторизация.
77. Разграничение доступа к объектам ОС. Домены безопасности. Матрица доступа
78. **РЕСУРСЫ**
    1. **Основная литература**
    2. Карпов В. Е., Коньков К. А. Основы операционных систем. – М.: ИНТУИТ.РУ «Интернет-университет информационных технологий», 2011 (или другие годы изданий) — 536с. Электронные версии в свободном доступе:

* <http://www.intuit.ru/studies/courses/2192/31/info> — лекционный курс
* <http://www.intuit.ru/studies/courses/1088/322/info> Видео — лекционный курс.
* <http://www.intuit.ru/studies/courses/2249/52/info> - практические занятия.
  1. **Дополнительная литература**

1. Столингс В. Операционные системы. – М.: Издательский Дом «Вильямс», 2001.
2. Олифер В.Г., Олифер Н.А. Сетевые операционные системы. – СПб.: Питер, 2001.
3. Стивенс У. UNIX: Взаимодействие процессов. – СПб.: Питер, 2002.
4. Стивенс У. UNIX: Разработка сетевых приложений. – СПб.: Питер, 2003.
   1. **Программное обеспечение**

|  |  |  |
| --- | --- | --- |
| **№ п/п** | **Наименование** | **Условия доступа** |
| 1. | Операционная система Linux | Свободно распространяемое программное обеспечение |

* 1. **Профессиональные базы данных, информационные справочные системы,   
     интернет-ресурсы (электронные образовательные ресурсы)**

* 1. **Материально-техническое обеспечение дисциплины**

Учебные аудитории для лекционных занятий по дисциплине обеспечивают использование и демонстрацию тематических иллюстраций, соответствующих программе дисциплины в составе:

мультимедийный проектор с дистанционным управлением.

Учебные аудитории для лабораторных и самостоятельных занятий по дисциплине оснащены ­­­­­­­­­­­­­­­­­­­­­­­­ компьютерами с установленной операционной системой Linux.