сОСи 2. Вопросы из программы дисциплины 8.2

Молодые люди, хватит гоготать (с) Набебин А.А.
Не хихикать!!! (с) Набебин А.А.
Паша, холотропное дыхание (с) Шершаков С.А.

Учебник:
https://vk.com/doc408807987_441963988

Список вопросов (начиная со страницы **19**, пункт **8.2**):
https://vk.com/doc253124210_446662792?hash=f04ecbcfc29edab498&dl=939b18a115f0813acb

Итак, нам нужно кратко ответить на вопросы, осветить основные, моменты касающиеся этой темы.

- отвечать в этом доке? ДА пизда

Катя	**1 15 29** 43 57 71
Никита	2 16 44 58 72
Миша Ш.	3 17 31 59 73
Паша	4 18 32 46 74
Рома	5 19 33 47 61
Кэр	6 34 48 62 76
Саша	7 21 35 49 63 77
Егор	8 22 36 50 64
Андрей	9 23 37 51 65
Киря	10 24 38 52 66
Даня	11 25 39 53 67
Вьет	12 26 40 54 68
Миша С.	13 27 41 55 69
Федя	14 28 42 56 70
Леша	20 30 45 60 75 20

1. Понятие о вычислительном комплексе. Системное программное обеспечение и операционные системы.

(Про вычислительный комплекс не нашла, но думаю, что это то же самое что и вычислительная система)

Из чего состоит любая вычислительная система? **Во-первых**, из того, что в англоязычных странах принято называть словом hardware, или техническое обеспечение: процессор, память, монитор, дисковые устройства и т.д., объединенные магистральным соединением, которое называется шиной.

Во-вторых, вычислительная система состоит из программного обеспечения. Все программное обеспечение принято делить на две части: прикладное и системное. К прикладному программному обеспечению, как правило, относятся разнообразные банковские и прочие бизнес-программы, игры, текстовые процессоры и т. п. Под системным программным обеспечением обычно понимают программы, способствующие функционированию и разработке прикладных программ.

Существует много точек зрения на то, **что такое операционная система**. Невозможно дать ей адекватное строгое определение. Операционная система предназначена для управления всеми частями весьма сложной архитектуры компьютера. операционная система – это программа, постоянно работающая на компьютере и взаимодействующая со всеми прикладными программами.

11. Критерии планирования и требования к алгоритмам планирования.

12.Параметры планирования.

Все параметры планирования можно разбить на две большие группы: статические параметры и динамические параметры. Статические параметры не изменяются в ходе функционирования вычислительной системы, динамические же, напротив, подвержены постоянным изменениям.

К статическим параметрам вычислительной системы можно отнести предельные значения ее ресурсов (размер оперативной памяти, максимальное количество памяти на диске для осуществления свопинга, количество подключенных устройств ввода-вывода и т. п.). Статические параметры процесса - кем запущен, степень важности, запрошенное процессоров время, какие требуются ресурсы и т.д.

Динамические параметры системы описывают количество свободных ресурсов на данный момент. Динамические параметры процесса – текущий приоритет, размер занимаемой оперативной памяти, использованное процессорное время и т.д.

Деятельность любого процесса можно представить как последовательность циклов использования процессора и ожидания завершения операций ввода-вывода. Промежуток времени непрерывного использования процессора носит название **CPU burst**, а промежуток времени непрерывного ожидания ввода-вывода – **I/O burst**.

15. Гарантированное планирование, приоритетное планирование

При **приоритетном** планировании каждому процессу присваивается определенное числовое значение – приоритет, в соответствии с которым ему выделяется процессор. Процессы с одинаковыми приоритетами планируются в порядке FCFS (First Come First Services). Для алгоритма SJF (Shortest Job First) в качестве такого приоритета выступает оценка продолжительности следующего CPU burst. Чем меньше значение этой оценки, тем более высокий приоритет имеет процесс. Для алгоритма гарантированного планирования приоритетом служит вычисленный коэффициент справедливости. Чем он меньше, тем больше у процесса приоритет.

Совершенно иной подход к планированию заключается в предоставлении пользователям реальных обещаний относительно производительности, а затем в выполнении этих обещаний (**гарантированное планирование**). Одно из обещаний, которое можно дать и просто выполнить, заключается в следующем: если в процессе работы в системе зарегистрировано n пользователей, то вы получите 1/n от мощности центрального процессора. Аналогично этому, в однопользовательской системе, имеющей я работающих процессов, при прочих равных условиях каждый из них получит 1/я от общего числа процессорных циклов. Это представляется вполне справедливым решением.

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

- Создание процесса - завершение процесса
- Запуск процесса (*готовность -> исполнение*) - приостановка процесса (*исполнение -> готовность*)
- Изменение приоритета процесса: нет пары

Ответ: изменение приоритета процесса

26. Нити исполнения и их отличие от процессов.

Усилия, направленные на ускорение решения задач в рамках классических операционных систем, привели к появлению новой абстракции внутри понятия

"процесс" – нити исполнения или просто нити. Нити процесса разделяют его программный код, глобальные переменные и системные ресурсы, но каждая нить имеет собственный программный счетчик, свое содержимое регистров и свой стек. Теперь процесс пред- ставляется как совокупность взаимодействующих нитей и выделенных ему ресурсов. Нити могут порождать новые нити внутри своего процесса, они имеют состояния, аналогичные состояниям процесса, и могут переводиться операционной системой из одного состояния в другое. В системах, поддерживающих нити на уровне ядра, планирование использования процессора осуществляется в терминах нитей исполнения, а управление остальными системными ресурсами – в терминах процессов. Накладные расходы на создание новой нити и на переключение контекста между нитями одного процесса существенно меньше, чем на те же самые действия для процессов, что позволяет на однопроцессорной вычислительной системе ускорять решение задач с помощью организации работы нескольких взаимодействующих нитей.

29. Понятие критической секции процесса

Критическая секция – это часть программы, исполнение которой может привести к возникновению race condition для определенного набора программ. Чтобы исключить эффект гонок по отношению к некоторому ресурсу, необходимо организовать работу так, чтобы в каждый момент времени только один процесс мог находиться в своей критической секции, связанной с этим ресурсом. Иными словами, необходимо обеспечить реализацию взаимоисключения для критических секций программ. Реализация взаимоисключения для критических секций программ с практической точки зрения означает, что по отношению к другим процессам, участвующим во взаимодействии, критическая секция начинает выполняться как атомарная операция.

30. Задача про процессы

. Пусть в вычислительную систему поступают пять процессов различной длительности с раз- ными приоритетами по следующей схеме:

Номер процесса	Момент поступления в систему	Время испол- нения	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.

Решение:

Захерачим таблицу:

Процесс /Время	1	2	3	4	5	6	7	8	9	10
1				**И**	**И**	**И**	Г	Г	Г	Г
2							**И**	**И**	**И**	**И**
3	**И**	**И**	**И**	Г	Г	Г	Г	Г	Г	Г
4			Г	Г	Г	Г	Г	Г	Г	Г
5					Г	Г	Г	Г	Г	Г

Процесс /Время	11	12	13	14	15	16	17	18	19	20
1	**И**	**И**	**И**	**И**	**И**	**И**	**И**			
2										
3	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г
4	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г
5	Г	Г	Г	Г	Г	Г	Г	**И**	**И**	**И**

Процесс /Время	11	12
1		
2		
3	**И**	
4	Г	**И**
5		

Процесс 1: 15
Процесс 2: 5
Процесс 3: 12
Процесс 4: 11
Процесс 5: 17

Бля, ну я хз

45. Термин «критическая секция» относится:

к участку процесса, выполнение которого совместно с другими процессами может привести к неоднозначным результатам.

54. Систематизация внешних устройств и интерфейс между базовой подсистемой ввода-вывода и драйверами

Устройства обычно принято разделять по преобладающему типу интерфейса на следующие виды:

- символьные (клавиатура, модем, терминал и т. п.);
- блочные (магнитные и оптические диски и ленты, и т. д.);
- сетевые (сетевые карты);
- все остальные (таймеры, графические дисплеи, телевизионные устройства, видеокамеры и т. п.);

Символьные устройства обычно умеют совершать две общие операции: ввести символ (байт) и вывести символ (байт) – get и put.

Для блочных устройств, таких как магнитные и оптические диски, ленты и т. п. естественными являются операции чтения и записи блока информации – read и write, а также, для устройств прямого доступа, операция поиска требуемого блока информации – seek.

Драйверы символьных и блочных устройств должны предоставлять базовой подсистеме ввода-вывода функции для осуществления описанных общих операций. Помимо общих операций, некоторые устройства могут выполнять операции специфические, свойственные только им. Для выполнения таких специфических действий в интерфейс между драйвером и базовой подсистемой ввода-вывода обычно входит еще одна функция, позволяющая непосредственно передавать драйверу устройства произвольную команду с произвольными параметрами, что позволяет задействовать любую возможность драйвера без изменения интерфейса. В операционной системе Unix такая функция получила название ioctl (от input-output control).

Помимо функций read, write, seek (для блочных устройств), get, put (для символьных устройств) и ioctl, в состав интерфейса обычно включают еще следующие функции.

• Функцию инициализации или повторной инициализации работы драйвера и устройства – open.

• Функцию временного завершения работы с устройством (может, например, вызывать отключениеустройства) – close.

• Функцию опроса состояния устройства (если по каким-либо причинам работа с устройством производится методом опроса его состояния, например, в операционных системах Windows NT иWindows 9x так построена работа с принтерами через параллельный порт) – poll.

• Функцию останова драйвера, которая вызывается при останове операционной системы или выгрузке драйвера из памяти, halt.

60. Что понимается под термином «внешняя фрагментация»?

потеря части памяти, не выделенной ни одному процессу.

68. Многоуровневая модель построения сетевых вычислительных систем.

Самым нижним уровнем в слоеных сетевых вычислительных системах является уровень, на котором реализуется реальная физическая связь между двумя узлами сети. Из предыдущего раздела следует, что для обеспечения обмена физическими сигналами между двумя различными вычислительными системами необходимо, чтобы эти системы поддерживали определенный протокол физического взаимодействия – горизонтальный протокол. На самом верхнем уровне находятся пользовательские процессы, которые инициируют обмен данными. Количество и функции промежуточных уровней варьируются от одной системы к другой.

В сетевых вычислительных системах все их одинаковые уровни, лежащие выше физического, виртуально обмениваются данными посредством горизонтальных протоколов. Наличие такой виртуальной связи означает, что уровень N компьютера 2 должен получить ту же самую информацию, которая была отправлена уровнем N компьютера 1. Хотя в реальности эта информация должна была сначала дой- ти сверху вниз до уровня 1 компьютера 1, затем передана уровню 1 компьютера 2 и только после этого доставлена снизу вверх уровню N этого компьютера.

Формальный перечень правил, определяющих последовательность и формат сообщений, которыми обмениваются сетевые компоненты различных вычислительных систем, лежащие на одном уровне, мы и будем называть сетевым протоколом.

Всю совокупность вертикальных и горизонтальных протоколов (интерфейсов и сетевых протоколов) в сетевых системах, построенных по "слоеному" принципу, достаточную для организации взаимодействия удаленных процессов, принято называть семейством протоколов или стеком протоколов. Сети, построенные на основе разных стеков протоколов, могут быть объединены между собой с использованием вычислительных устройств, осуществляющих трансляцию из одного стека протоколов в другой, причем на различных уровнях слоеной модели.

75. Идентификация и аутентификация.

В рот ебал эти оси