лабораторная 1
1 часть обработчик прерывания int 8h мы дизасемблируем код с помощью программы получаем листинг и по полученным кодам строим схемы алгоритмов

lgmd - написать вместо двигателя
спрашивает про команду lock

счетчик тиков, а не сетки прерываний

2 часть изучения функций обработчика прерываний от системного таймера 2х семейств ОС для windows и unix-linux + изучение каким образом в этих семействах ОС пересчитываются динамические приоритеты (меняются в процессе выполнения программы, могут быть только у пользовательских программ (у приложений))

оформление:
отдельно указываем функции обработчика прерываний от системного таймера для винды и юникс
при этом выделяем:
1. функция обработчика прерывания которую он выполняет: по тику, по главному тику и по кванту

тик - частота 18,2 гц это выделенный импульс из тактовой частоты, независимое от процессора действие

в защищенном режиме тики другие

главный тик - несколько тиков
главных тиков в системе может быть несколько

обработчик прерывания от системного таймера (ОПОСТ) выполняется на очень высоком уровне приоритета
когда он выполняется никакая другая работа в системе выполняться не может

ОПОСТ должен завершаться как можно быстрее - то единственное быстрое прерывание в системе

пересчетом приоритетов занимается специальная часть кода ядра которая называется планировщик

когда истекает квант, то выделение процессу находящемуся первому в очереди к процессору за получением процессорного времени занимается часть кода ядра которая называется диспетчер

ОПОСТ не может вызывать функции
он может только инициировать действия
они будут выполняться после того как ОПОСТ завершится

windows
эти отложенные действия выполняются через DPC

unix
инициализация отложенных действий выполняется или установкой флага или изменением статуса …..

в результате поток ставится в очереди процессору в соответствии со своим приоритетом

квант выделяется процессу который стоит первым в очереди готовых процессов т.е. имеет наивысший приоритет

время процессов измеряется квантами

Пересчет приоритетов:
в обоих семействах ОС учитывается ожидание процесса в очереди готовых процессов
делается для того чтобы исключить **бесконечное откладывание** процессов

в конце отчета нужен вывод СУДЬБОНОСНЫЙ!!!

книги:
соломон русинович windows мастер-класс
вахалия unixx изнутри

в каком режиме выполняется привилегированная команда GDTR и о чем это говорит
смещение не может превышать FFFF
прочитать про теневые регистры (будут спрашивать)

2 лаба
перевод компьютера из реального режима в защищенный
в защищенном режиме …. (остальное в тетради)
корректно - используя 32х разрядные операнды

книги зубков, рудаков программирование на ассемблере

1. перевод компьютера из реального режима в защищенный
во всех режимах выводить на экран какой сейчас режим
2. определение объема доступного адресного пространства в защещенном режиме
в защищенном режиме 1й МБ пропускаем (почему?)
начиная со 2го МБ считываем байт памяти, записываем его в сигнатуру, сравниваем со своей сигнатурой, если они совпали, то это память
сигнатура - код который мы сами назначаем
полученный объем памяти в 10й системе счисления вывести на экран
3. работа с прерываниями
от системного таймера и от клавиатуры
должны написать 2 обработчика прерывания
создать таблицу для дескрипторов

3 лабы на linux:
3я лаба командная строка unix
ps - выводит информацию о процессах
ls - выводит информацию о файлах

смотрим что время квантуется
изучаем линки
изучаем именованные программные каналы

ls -al
9 букв - права доступа
1-3 буквы права доступа юзера
4-6 три группы юзеров

столбик чисел хард линки (жесткая ссылка)(еще одно равноправное имя файла)
и оникс имя файла не является его идентификатором
один и тот же файл в системе может иметь много имен
идентификатором файла является inode
индентификатором файлов в системе является номер inode
номер inode в системе является методанными

надо создать хард линк
на свои файлы
смотрим что файлы разные, а inode одинаковый

inode - дескриптор файла содержащие информацию необходимые для доступа к информации хранящийся в файле

файл - любая поименованная совокупность данных хранящаяся во вторичной памяти

софт линк - специальный файл который содержит строку
и эта строка является путем к файлу

вопросы:

1. **ps -al что это (коротко назвать назначение информации которую мы видим на экране, по полям , их смысл)**
2. **флаги, какие флаги определены в системе, что они означают**
3. **что значат права суперюзера**
4. **почему для системы важно : после форка был экзэк или после форка его не было**
5. **перечислить состояния, их смысл, что они означают**
6. **почему блокировку на вводе выводе нельзя прервать (исходить из собственной практики программирования)**
7. **кто такой зомби**
8. **почему для пережат флаг 0, а для чаилда 1** (потому что мы запустили новую программу) (мы вызвали программу из командной строки, был предок мы вызвали новую программу поэтому был fork и exec) (для нашей программы предок это терминал)
9. **какие процессы называются процессами сиротами, какие действия выполняет система**
10. **при ней изменить приоритет текущего процесса**
11. **почему без прав суперюзера мы можем только понизить приоритет процессора** (потому что это влияет на планирование) (если каждый пользователь будет повышать приоритет то возникнет неразбериха) (повышение приоритета означает - постановка в начало очереди)
12. **какую информацию показывает whn**

13. la = al
14. какие типы файла различает система
15. 9 букв какие права доступа перечисляют
16. права доступа что показывают
17. надо создать жесткую ссылку программный канал
18. передать сообщения между консолями
19. показать номер inoda

потомок не знает что у предка есть wait

зачем нужны права суперюзера

4я лаба 5 основных системных вызовов linux:
fork {}
exec {}
wait {}
pipe {}
signal {}

в unix программа запускается в 2 этапа:
1. создается новый процесс
2. процесс потомок переходит на новое адресное пространство

исполняемые файлы:
1. исходные (.asm)
2. объектный (.obj)
3. исполняемый (.exe, а в unix - .o)

5 лаба взаимодействие параллельных процессов
пишем 2 программы
1. на основе алгоритма производства потребления
2. алгоритма читателя писателя

6 лаба задача читатели писатели монитор хоара под windows
реализуем задачу на параллельных потоках и eventax

должно быть 3 счетчика: ждущих писателей, ждущих читателей, активных читателей

топ требования: включить в код 1 мьютекс
задачу реализует на одиночной переменной

критическая секция читателя открывается функций stopread, закрывается stopread, не надо между ними ставить мьютекс

мьютекст должен захватываться и освобождаться в одной функции

надо исползовать inter loked increment / decrement - использовать для работы

ошибки которые не стоит повторять:
- нельзя блокировать всю базу, блокирует только конкретное поле (запись)
- не нужен оконный интерфейс, нужно чтобы было при выполнении видно: какой поток что записал, какой поток что прочитал
- не надо вводить случайное значение разделяемой переменной, разделяемую переменную процессы строго инкрементируют
-