Воеводин ВМК МГУ про обработку информации
параллельные вычисления

24/02/2025

Вступление

1 лаба - bash/скрипты
2 лаба - СЛАУ с разреженной матрицей
разные методы
3 лаба - половина семестра

Взаимодействие между процессами. Клиент-сервер
ПО на TCP-сокетах
Проект не использует взаимодействие в среде bash - отдельно защитить
параллельных вычислений нет - защитить отдельно
1 допуск к сессии - автомат
Написать на бумажке задачка в параллельном сегменте с помощью чего-то. 10-15 строк кода
bash

Начало

Операционные системы
 123
Большинство операционных систем базируются на семействе операционных систем типа UNIX.
Список популярных операционных систем, базирующихся на UNIX:
BSD, Linux, MacOS

Задачи современных операционных систем:

1. Мультизадачный режим работы
   #Одновременность - 2 и более программы работают одновременно, если временные интервалы их выполнения пересекаются хотя бы частично.
2. Управление устройствами ввода-вывода
3. Управление оперативной памятью
4. Взаимодействие процессов
5. Разграничение полномочий

Лекция 2. Мультизадачность

1. Одновременное исполнение нескольких задач
   Важное понятие одновременности. goto: Одновременность
2. Пакетный режим
   #Пакетный_режим - это такой способ определения мультизадачности, при котором смена активной задачи происходит только в случае её окончания или запроса на операцию ввода-вывода.

3. Режим разделения времени
   #Режим_разделения_времени - каждой задачи отводится время работы, называемое квантом времени. Он генерируется таймером, инициирующим прерывания через определённые промежутки времени - кванты.
4. Планирование времени центрального процессора в режиме реального времени
   #RTOS делятся на 2 вида:
   мягкие и жёсткие
   В мягкой системе если процесс не может своевременно ответить на некий запрос / выполниться в нужный момент времени, то он блокируется на какой-то промежуток времени
   В жёсткой системе в случае если процесс не может отреагировать на запрос, то этот процесс останавливается
   Qnx
5. Требования к аппаратуре для обеспечения многозадачности
   Аппарат прерывания. Программная и аппаратная часть.

Защита памяти с ...

10/03/2025

В пятницу дополнительное занятие до 2?
Удалённнный формат

У программы есть возможность влиять только на свою память.

Привилегированный и ограниченный режим центрального процессора
Пользовательские программы работают в ограниченном режиме работы центрального процессора - набор действий: взаимодействие с памятью и запрос на использование системных ресурсов.
Привилегированный - только ядро и больше ничего.
В этом режиме допустимы команды для работы с внешними устройствами, взаимодействие с памятью.

Таймер - устройство, генерирующее прерывания через равные промежутки времени.

Для того чтобы мультизадачный режим работал нужны:
Аппарат прерываний
Защита памяти
Привилегированный режим.
Для пакетного режима этого достаточно, но для разделения времени нужен таймер.

Какие бывают прерывания:
Внешние прерывания (аппаратные прерывания)

1. Устройство, которому нужно внимание процессора устанавливает на шине запрос прерывания
2. Процессор доводит выполнение текущей программы до точки, в которой процесс её работы можно прервать (чтобы потом с этой точки процесс можно было продолжить). После приостановки процессор выдает сигнал подтверждения прерывания
3. После получения сигнала устройство передаёт число-идентификатор - номер прерывания. Процессор сохраняет в стеке текущее состояние счётчика команд. (Малое упрятывание???)

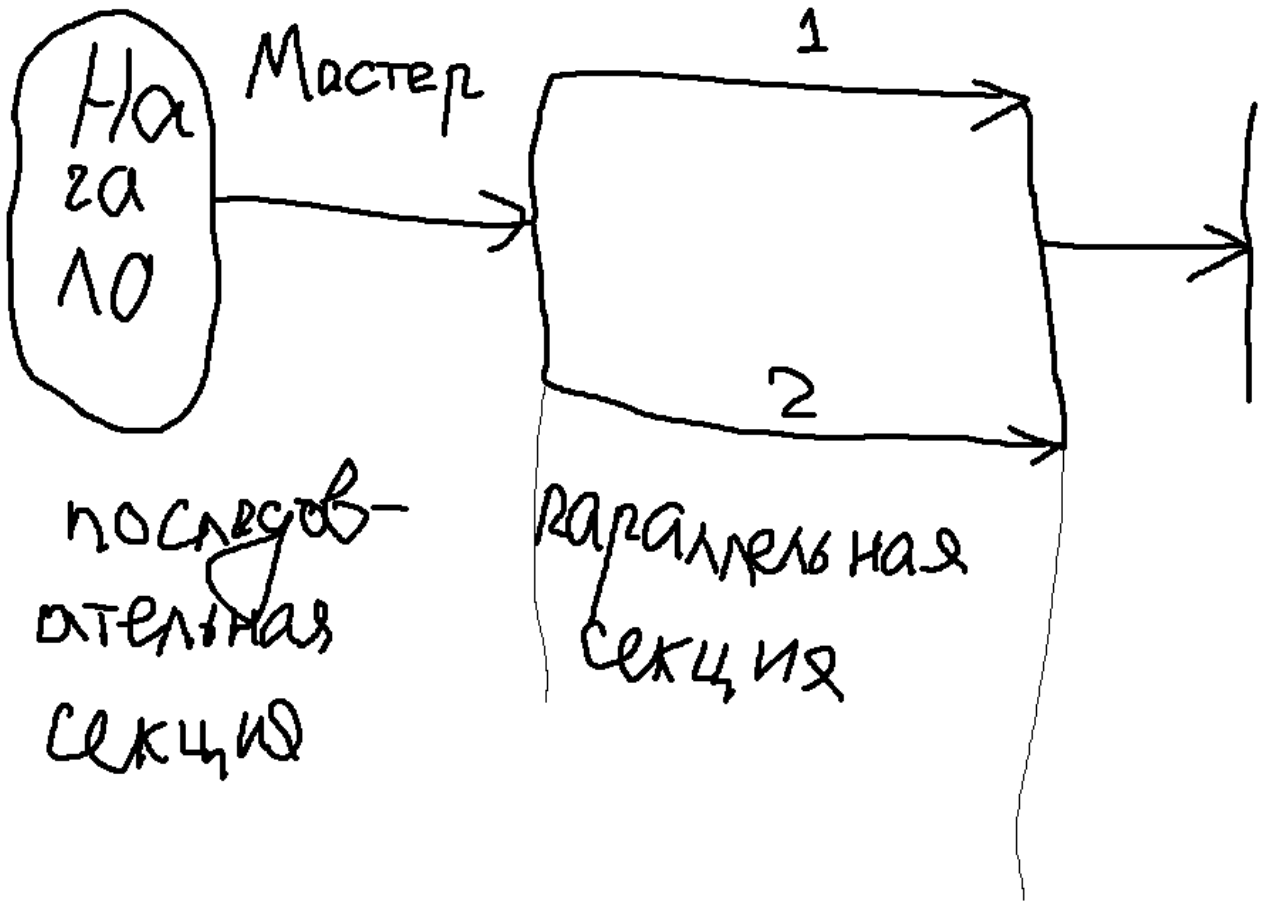
4. Устанавливается привилегированный режим работы процессора. (Потому что нужно обработать внешнее устройство)
5. Выполняется обработчик прерываний

Прерывание - известный способ переключения процессора в привилегированный режим.
Внутренние прерывания (ловушки) (программные прерывания)
Программные прерывания и системные вызовы.
Обращение пользовательской программы или процесса к ядру операционной системы за услугами называется системным вызовом. (Передача управления от программы к ядру).
Программное прерывание - прерывание, вызванное по инициативе пользователя.
Привилегированный и ограниченный режим работы процессора.
В основе ОС всегда находится программа, осуществляющая работу с аппаратурой, обрабатывающая прерывания и системные вызовы - ядро (kernel).
Никакой код никакого пользовательского процесса ни при каких условиях не может быть исполнен в привилегированном режиме.
Эмуляция физического компьютера.
С эмуляцией другой архитектуры сложно.

О второй лабе:
Поток в центральном процессоре и поток о котором мы будем говорить - разные вещи
Любая программа ничинается с точки входа.
По умолчанию у программы 1 поток - мастер.
Поток - последовательность выполнения комманд.
Файловый менеджер, когда копирует данные:
Создать переменную типа int и считаем сумму
openmp
<omp.h>
Требования:
результат работы программы не зависит от количества потоков

```
int sum
#pragma omp parallel
{
sum = sum + v[i];
}
cout << ... << ...;
#pragma omp for
```

Создаём указатель из миллиона 1



24/03/2025

1 матрица хранится построчно
2 матрицу можно хранить по столбцам, а не по строчкам (0_o)
Для того чтобы обеспечить оптимальную производительность (минимальный простой процессора) нужно не только разбить работу на потоки, но ещё разбить разбить работу так, чтобы память у потоков была минимальной

Для разреженной матрицы:
IJA
строка-столбец -ненулевой элемент
I ptr I0, I1, I*n*
J ptr - J_0, J_1, J_k начало строки
A - ненулевое значение, соответствующее столбцу
A[k*k]
CSR

Блочный способ хранения
Матрица n на n блоков
Каждый блок имеет размерность пусть 8 на 8
B1 1 блок - массив A из ненулевых элементов
наличие ненулевого элемента - 1.
для блока в 64 элемента: 8 байт

1 Разбиваем таблицу на блоки (квадратные)
2 У каждого блока есть свой номер. Храним координаты ненулевых блоков

1 Кэш
2 Мультипоток
3 Расширенные инструкции

ijk	**1.8**
ikj	1.5
avx:	0.03
avx без openmp	0.15
ijk без openmp	4.61

152 раза

Операции на векторах
Посмотреть спецификацию процессора
8 значений за 1 такт

123 операции схожи, но с разными данными
1 поток $I_1^1 I_2^1 I_3^1$
2 поток $I_1^2 I_2^2 I_3^2$
...
С avx:
P1 P2 P3
где P - это вектор-столбец
Пусть d = a+b*c
Процедура mut = a+b*c

Допустим, создаем класс, где хранятся массивы
Если существуют часто вместе используемые массивы их нужно хранить вместе
Это повысит вероятность попадания в кеш
Компилятор может попытаться догадаться даже до avx mtx-файл

Запоминающие устройства
Стриммер гораздо медленнее жёсткого диска
Диски
ОЗУ
Кэш
Регистр SSE,AVX

Управление оперативной памятью
Менеджер памяти
Проблемы, которые решает менеджер памяти:
1 Защита процессов друг от друга и операционной системы от процессов. Управление аппаратной защитой памяти

2 Нехватка оперативной памяти. (засчёт подкачки жесткого диска)
3 Дублирование данных.
4 Перемещение кода.
5 Фрагментация.

21/04/2025

Рассматриваем виртуальную память.
База и предел - каждой программе выделяется блок физической памяти.
Адрес ячейки = виртуальный адрес + сдвиг
Сегментная модель - модель организации виртуальной памяти.
Вся память разделяется на блоки одинакового размера (сегменты), каждой программе выделяется целое количество блоков.

Где-то есть таблица с номером сегмента, начало сегмента
Чтобы узнать физический адрес селектор должен найти строку в таблице
Физический адрес = виртуальный адрес (аггрегирование) номер сегмента (аггрегирование) База

Сильные и слабые стороны сегментной:
Плохо обрабатывает легковесные процессы.

Страничная организация памяти.
Всю физическую память разделили на блоки - кадры.
Кадры делятся на страницы
Виртуальный адрес->Номер страницы->смещение
| |
\/ \/
Номер кадра -> смещение
\ /
|
\/
Физический адрес

valloc

Клиент-серверная модель
Клиент - программа, отправляющая запросы.
Сервер - программа, отвечающая на запросы.

Для того, чтобы организовать взаимодействие программ, создали API.
Сокет - объект ядра операционной системы, через который происходит сетевое взаимодействие. Для идентификации сокетов используются адреса, они могут быть разные. 2 базовых подхода:
TCP-IP v4 - 4 числа от 0 до 255.
Адрес позволяет идентифицировать компьютер в сети. Порт нужен для идентификации программы.
IP v6 - 8 чисел в hex.
Протокол (обмена) - набор соглашений, которому должны следовать участники обмена информацией (чтобы обмениваться информацией).
Пример: программы находятся на разных физических машинах.
Они должны:

Иметь возможность физически регистрировать сигналы друг от друга
Понимать, как интерпретировать сигналы в качестве информации
INTERNATIONAL STANDARD ORGANISATION (ISO)
OPEN SYSTEM INTERCONNECTION (OSI) - стандарты связи.
Протокол, состоящий из 7 уровней:
Физический - набор соглашений физического соглашения между машинами (число проводов, частота сигнала, прочие характеристики)
Модель включает семь уровней:
1 Физический. Соглашения об использовании физического соединения между машинами, включая количество проводов в кабеле, частоту и другие характеристики сигнала, и т.п.
2 Канальный. Соглашения о том, как будет использоваться физическая среда для передачи данных; это включает, например, коррекцию ошибок
3 Сетевой. Компьютерная сеть обычно состоит из множества каналов, соединяющих компьютеры. Некоторые компьютеры, называемые шлюзами, подключаются к нескольким каналам одновременно и передают пакеты из одного канала в другой. Сетевой уровень протоколов определяет соглашения о том, как данные будут передаваться по сети, так, чтобы из любой точки сети можно было передать данные в любую другую точку сети, вне зависимости от того, по скольким каналам придется передавать информацию, и сколько шлюзов при этом будет задейство
вано. Сетевой уровень отвечает за адресацию, маршрутизацию пакетов и т.п.
4 Транспортный. Пакеты, передаваемые по сети с помощью протоколов сетевого уровня, обычно ограничены в размерах и, кроме того, могут доставляться не в том порядке, в котором были отправлены, теряться, а в некоторых случаях искажаться. Обычно прикладным программам требуется более высокий уровень сервиса, обеспечивающий надежность доставки данных и простоту работы. За это отвечают протоколы транспортного уровня; реализующие их программы сами следят за доставкой пакетов, отправляя и анализируя соответствующие подтверждения, ну
меруют пакеты и рассатвляют их в нужном порядке после получения, и т.п.
За реализацию этого уровня отвечают специальные программы в ОС.
5 Сеансовый. Определяет порядок проведения сеанса связи, очередность запросов и т.п.
6 Представительный. На этом уровне определяются правила представления данных, в частности, кодировка, правила представления двоичных данных текстом, и т.п.
7 Прикладной. Протоколы этого уровня определяют, как конечные приложения будут использовать соединения для решения конкретных задач, для которых они предназначены.

Возможную потерю сокетов стоит учитывать.
22 порт - ssh
Лучше не использвовать порты ниже 1000

Сокеты буывают по типу взаимодействия 2 типов:
1 - потоковый сокет
2 - датаграммный

Потоковый тип взаимодействия предоставляет прикладному программисту **иллюзию** надёжного двунаправленного канала передачи данных. Данные могут быть записаны порциями любого размера (нелюбого, зависит от ОС). Большие лучше делить на маленькие. Гарантируется, что данные на другом конце будут получены в нужном порядке и без потерь, либо не получены вообще (ошибка).
При датаграммном соединении доступны 2 операции: передача пакета и приём (с проверкой есть ли он) пакета. При этом пакеты могут не прийти - никто не гарантирует, что пакеты дойдет, что пакеты не

могут быть задублированны.
Целостность, не прийти, дублирование, разный порядок.
Данные должны быть отправлены пакетами установленных размеров.
Контроль целостности пакетов и обработка ошибок ложится на плечи пользователя (программиста)

Кому-то достанется потоковый, кому-то датаграммный