|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |

**Практические МАТЕРИАЛЫ №4**

|  |  |
| --- | --- |
| **Операционные системы** | |
| *(наименование дисциплины (модуля) в соответствии с учебным планом)* | |
| Уровень | бакалавриат |
|  | *(бакалавриат, магистратура, специалитет)* |
| Форма обучения | очная |
|  | *(очная, очно-заочная, заочная)* |
| Направление(-я)  подготовки | 09.03.02 Информационные системы и технологии |
|  | *(код(-ы) и наименование(-я))* |
|  |  |
| Институт | комплексной безопасности и специального приборостроения |
|  | *(полное и краткое наименование)* |
| Кафедра | КБ-4 «Интеллектуальные системы информационной безопасности» |
|  | *(полное и краткое наименование кафедры, реализующей дисциплину (модуль))* |
| Лектор | Латыпова Ольга Валерьевна |
|  | *(сокращенно – ученая степень, ученое звание; полностью – ФИО)* |

|  |  |  |  |
| --- | --- | --- | --- |
| Используются в данной редакции с учебного года | 2020/2021 | | |
|  | *(учебный год цифрами)* | | |
| Проверено и согласовано «\_\_\_\_» \_\_\_\_\_\_\_\_20\_\_\_г. |  |  |  |
|  | *(подпись директора Института/Филиала с расшифровкой)* | | |

Москва 2020 г.

Практическое занятие 5, 6

Задание

1. Разработать (выбрать уже готовые ) три программных модуля запускаемых в отдельном потоке каждый. Потоки не бесконечны, и завершаются не одновременно.

2. разработать планировщик задач, запускающий потоки на выполнение из очереди потоков, согласно схеме.

Реализация планировщика с тремя состояниями:

ГОТОВ 🡨🡪ВЫПОЛНЕНИЕ 🡪ОЖИДАНИЕ

ОЖИДАНИЕ

блокировка

событие

пробуждение

ГОТОВ (очередь)

ВЫПОЛНЯЕТСЯ

порождение

истечение кванта

выполнен

запуск

В качестве ожидаемого события может быть выбрано нажатие определенной клавиши.

Например: Поток блокируется по нажатию определенной клавиши.

(+ / -) – (увеличение / уменьшение) приоритета.

Пробуждение – по нажатию той же клавиши.

**Общая схема работы**

1. запуск планировщика
2. инициализация потоков и постановка их в очередь готовых к выполнению
3. выбор первого потока из очереди готовых
4. обращение к контексту потока и запуск потока
5. в каждом элементарном цикле обработка счетчика тиков и проверка истечения кванта:

если квант закончился – передача управления планировщику, нет продолжение работы потока

1. для прерванного потока запоминается контекст, а поток ставится в очередь готовых
2. если очередь не пуста – возврат на 3, иначе завершение программы

**Требования к реализации (для всех вариантов)**

* три программных модуля + планировщик
* наличие структуры, имитирующей БУП
* обязательно инициализируемая в начале работы очередь готовых потоков

## Теоретическая часть

**1.1. Цель работы**

Ознакомление с основными принципами организации системы планирования процессов мультизадачной ОС, моделирование и программная реализация планировщика задач для мультизадачной операционной системы

**1.2. Теоретические сведения.**

Выполняемые компьютером под управлением ОС вычисления есть исполнение некоторых инструкций, инициируемое как пользовательскими процессами, так и операционной системой. При этом управление многократно передается от ОС пользовательскому процессу и наоборот. Инструкции, предназначенные для исполнения, размещаются в ОП. Процессор, получив от операционной системы команду на обращение за инструкциями к определенной ячейке памяти, начинает последовательно исполнять эти инструкции, выбирая их из ячеек, следующих за указанной. Последовательный выбор команд из смежных ячеек памяти происходит до тех пор, пока не возникает некое системное событие, которое процессор должен обработать. Тогда выбор команд прекращается, и процессор обращается за инструкциями к области памяти, в которой размещена некая системная программа, необходимая для обработки данного события, в таком случае говорят, что управление передается операционной системе. Такая организация вычислений позволяет довольно легко организовать обработку любого события в системе, необходимо только сделать так, чтобы процессор «заметил» произошедшее событие и чтобы системе планирования был доступен *блок управления процессом* обработки этого события (при этом кодовый сегмент, соответствующий процессу обработки должен находиться в памяти).

1. ***Описание рабочих структур.***

*Процесс* (или *задача*) ­ это не сама программа, а некая абстракция, описывающая выполняющуюся в вычислительной системе программу, как пользовательскую, так и системную. Материальным (информационным) представителем любого процесса в вычислительной системе является *блок управления процессом* (БУП). БУП содержит информацию, позволяющую многократно прервать, а затем продолжить выполнение процесса, восстанавливая состояние операционной среды, которое было в момент прерывания процесса. Фактически ОС оперирует не процессами, а их блоками управления.

БУП состоит из двух частей: дескриптор и контекст. *Дескриптор* процесса по сравнению с контекстом содержит более оперативную информацию, которая должна быть *легко доступна подсистеме планирования* процессов. *Контекст* процесса содержит менее актуальную информацию и используется операционной системой *только после того, как* *принято решение* о возобновлении прерванного процесса.

*Дескриптор* − это указатель на процесс; он содержит:

* уникальный идентификатор процесса,
* информацию о состоянии процесса,
* данные о степени привилегированности процесса (приоритет),
* место нахождения кодового сегмента,
* ссылка на контекст процесса,
* другую оперативную информацию, необходимую для принятия решения о возобновлении прерванного процесса.

*Контекст* − подробное описание процесса, которое отражает состояние операционной среды в момент прерывания; он содержит состояние операционной среды, отображаемое:

* состоянием регистров и программного счетчика,
* режимом работы процессора, указателями на открытые файлы,
* информацией о незавершенных операциях ввода-вывода,
* кодами ошибок выполняемых данным процессом системных вызовов и т.д.

Процессорная обработка программного кода может начаться только после того как для него операционной системой будет создан (порожден) соответствующий этой программе процесс. Порождение системой процесса означает:

* создание информационной структуры, описывающей процесс (дескриптор и контекст процесса).
* включение дескриптора процесса в очередь готовых процессов;
* загрузка кодовый сегмент процесса в оперативную память.

Формирование системой информационной структуры, соответствующей процессу, является очень важным этапом в жизненном цикле процесса. Наиболее существенным моментом здесь является уникальное именование процесса в системе, поскольку именно по этому уникальному системному имени (идентификатору) процесс будет распознаваться в любой ситуации. Наиболее простой и надежный способ формирования уникального идентификатора – пересчет процессов в момент инициализации (порождения) очередного процесса. Так в операционных системах семейства UNIX каждый порождаемый процесс получает свой собственный уникальный идентификационный номер PID (Process IDentificator). При создании нового процесса операционная система пытается присвоить ему свободный номер больший, чем у процесса, созданного перед ним. Если таких свободных номеров не оказывается (например, мы достигли максимально возможного номера для процесса), то операционная система выбирает минимальный из всех свободных номеров.

В однопроцессорной вычислительной системе исполняется (т.е. непосредственно обрабатывается процессором) всегда только один процесс. Процессов, готовых к исполнению, ожидающих и приостановленных может быть несколько. Очереди готовых, ожидающих и приостановленных процессов составляется из их дескрипторов. Очередь дескрипторов может быть организована как список дескрипторов, где каждый дескриптор имеет указатель на следующий за ним дескриптор (рис.1.1). Такая организация очередей позволяет легко их переупорядочивать, включать и исключать процессы, переводить процессы из одного состояния в другое.

Описатель процесса А

Ссылка

Описатель процесса B

Ссылка

Описатель процесса C

Ссылка

Описатель процесса D

Ссылка

Описатель процесса E

Ссылка

Рис. 1.1. Очередь процессов

И контекст, и дескриптор могут быть объединены в одну структуру, но могут быть и более сложные структуры, т.е. информация, относящаяся к контексту, может быть дополнительно сгруппирована, − тогда для одного процесса существует несколько контекстов.

1. ***Подсистема планирования – планировщик задач***

Задача краткосрочного планирования – *диспетчеризации*, заключается в реализации *динамического планирования* (on-line), когда решения принимаются во время работы системы на основе анализа текущей ситуации. В этом случае ОС все время работает в условиях неопределенности – процессы появляются в непредсказуемые моменты времени и так же непредсказуемо завершаются. Подсистема краткосрочного планирования (диспетчеризации) процессов является наиболее важной частью системы управления процессами, поскольку именно *планировщик* решает в системе задачу переключения с одного процесса на другой.

Диспетчеризация процессов сводится к следующему:

* сохранение контекста текущего процесса;
* загрузка контекста нового процесса, выбранного в результате планирования;
* запуск нового процесса на выполнение, т.е. передача управления этому процессу.

Функции планировщика:

* Постановка процессов в очередь на выполнение и управление этой очередью
* Выбор из очереди процесса и перевод его в активное состояние, т.е. передача ему контроля над центральным процессором.
* Распознавание системных событий и запуск соответствующих процессов-обработчиков

Постановка процессов в очередь и выбор из очереди процесса на исполнение осуществляется в соответствии с некоторой выбранной стратегией. Правила, в соответствии с которыми задача запускается на выполнение, называются *дисциплиной диспетчеризации* или *дисциплиной обслуживания (ДО)*. Алгоритм, реализующий правила обслуживания, называется *алгоритмом планирования процессов*. В простейшем случае это постановка в конец очереди и выбор из ее начала. Такая дисциплина обслуживания называется циклической или карусельной.

Передача управления от одного процесса другому включает в себя решение следующих задач:

* определение момента времени для смены выполняемого процесса;
* выбор процесса на выполнение из очереди готовых процессов;
* переключение контекстов "старого" и "нового" процессов.

При этом необходимо отметить, что сам планировщик не может отнять управление у одного процесса и передать другому. Если процесс по какой-то причине сам отдает управление, то управление переходит к планировщику, который выбирает из очереди готовых следующий процесс. Для того чтобы отнять управление у процесса принудительно, необходимо организовать некоторое системное событие, прерывающее выполнение процесса. Управление при этом также передается планировщику; планировщик определяет тип события и запускает соответствующий обработчик «переключением контекста». Если системное событие связано с окончанием времени обслуживания прерванного процесса – планировщик сохраняет текущий контекст для прерванного процесса, определяет следующий процесс на исполнение и запускает его опять же переключением контекста.

Среди множества различных алгоритмов планирования наиболее часто встречаются алгоритмы, основанные на *квантовании,* и алгоритмы, основанные на *приоритетах*.

*Квант* – это временной интервал, в течение которого процессу разрешено занимать процессор, т.е. разрешено находиться в состоянии ВЫПОЛНЕНИЯ. Понятие кванта основывается на периоде таймера, который называется *тиком* (~ 45мс). Квант равен целому числу *тиков* из промежутка [1,255].

В соответствии с алгоритмами, основанными на квантовании, смена активного процесса происходит, если:

* процесс покинул систему, поскольку завершился или произошла ошибка,
* процесс перешел в состояние ОЖИДАНИЕ,
* исчерпан квант процессорного времени, отведенный данному процессу.

Процесс, который исчерпал свой квант, переводится в состояние ГОТОВНОСТЬ и ожидает, когда ему будет предоставлен новый квант процессорного времени, а на выполнение в соответствии с определенным правилом выбирается новый процесс из очереди готовых. Кванты, выделяемые разным процессам, могут быть одинаковыми для всех процессов или различными. Кванты, выделяемые одному процессу, могут быть фиксированной величины или изменяться в разные периоды жизни процесса. Процессы, которые не полностью использовали выделенный им квант (например, из-за ухода на выполнение операций ввода-вывода), могут получить или не получить компенсацию в виде привилегий при последующем обслуживании.

Другая группа алгоритмов использует понятие "приоритет" процесса.

*Приоритет* - это число, характеризующее степень привилегированности процесса при использовании ресурсов вычислительной машины, в частности, процессорного времени: чем выше приоритет, тем выше привилегии.

Приоритет может выражаться целыми или дробными, положительным или отрицательным значением. Чем выше привилегии процесса, тем меньше времени он будет проводить в очередях. Приоритет может назначаться директивно администратором системы в зависимости от важности работы или внесенной платы, либо вычисляться самой ОС по определенным правилам; он может оставаться *фиксированным* на протяжении всей жизни процесса либо изменяться во времени в соответствии с некоторым правилом, т.е. быть *динамическим*. Приоритет может быть *относительным* – тогда управление у него принудительно отнято быть не может, и *абсолютным* – тогда при появлении в системе процесса с приоритетом большим, чем у выполняющегося процесса, управление принудительно передается другому процессу.

Роль планировщика в системе диспетчеризации процессов хорошо видна на рисунке 1.2, на котором показан фрагмент временной диаграммы работы планировщика в системе, где одновременно выполняются четыре процесса. В данном случае неважно, по какому правилу выбираются процессы на выполнение и каким образом изменяются их приоритеты. Существенное значение имеют лишь события, вызывающие активизацию планировщика.

Рис. 1.2. Моменты перепланировки процессов

**Т**

**ER**

***t***

**Т**

**Т**

**Т**

**Т**

**Т**

**Т**

**Т**

**Т**

**Т**

**Т**

**I/O**

**R**

**S**

**S**

**R**

Планировщик

Процесс 4

Процесс 2 2

Процесс 3

Процесс 1

Первые четыре цикла работы планировщика, приведенные на рисунке, были инициированы прерываниями от таймера по истечении квантов времени (эти события обозначены на рисунке как Т).

Следующая передача управления планировщику была осуществлена в результате выполнения процессом системного запроса на ввод-вывод (событие I/O). Планировщик перевел этот процесс в состояние ожидания, а затем переключил процессор на процесс 2. Процесс 2 полностью использовал свой квант, произошло прерывание от таймера, и планировщик активизировал процесс 1.

При выполнении процесса 1 произошло событие R – системный вызов, в результате которого освободился некоторый ресурс (например, был закрыт файл). Это событие вызвало перепланировку процессов. Планировщик просмотрел очередь ожидающих процессов и обнаружил, что процесс 4 ждет освобождения данного ресурса. Этот процесс был переведен в состояние готовности, но поскольку приоритет выполняющегося в данный момент процесса 1 выше приоритета процесса 4, планировщик вернул процессор процессу 1.

1. ***Переключение контекста***

Реально в однопроцессорной компьютерной системе в каждый момент времени может исполняться только один процесс. Для мультипрограммных вычислительных систем псевдопараллельная обработка нескольких процессов достигается с помощью переключения процессора с одного процесса на другой. Пока один процесс выполняется, остальные ждут своей очереди на получение процессора.

Как видим, каждый процесс может находиться как минимум в двух состояниях: *процесс исполняется* и *процесс не исполняется*. Диаграмма состояний процесса в такой модели изображена на рисунке 1.3.



Рис1.3. Простейшая диаграмма состояний процесса.

Процесс, находящийся в состоянии *процесс* *исполняется*, может через некоторое время завершиться или быть приостановлен операционной системой и снова переведен в состояние *процесс не исполняется.* Приостановка процесса происходит по одной из двух причин: для его дальнейшей работы потребовалось возникновение какого-либо события (например, завершения операции ввода-вывода) или истек временной интервал, отведенный операционной системой для работы этого процесса. После этого операционная система по определенному алгоритму выбирает для исполнения один из процессов, находящихся в состоянии *процесс не исполняется*, и переводит его в состояние *процесс исполняется.* Новый процесс, появляющийся в системе, первоначально помещается в состояние *процесс не* *исполняется*.

Для корректного переключения процессора с одного процесса на другой необходимо сохранить контекст исполнявшегося процесса и восстановить контекст процесса, на который будет переключен процессор. Такая процедура сохранения/восстановления работоспособности процессов называется *переключением контекста*. Время, затраченное на переключение контекста, не используется вычислительной системой для совершения полезной работы, называется *латентной задержкой* и представляет собой накладные расходы, снижающие производительность системы. Оно меняется от машины к машине и обычно находится в диапазоне от 1 до 1000 микросекунд.

1. ***Механизм прерываний***

*Прерывание* ­ принудительная передача управления от выполняемой программы к системе, происходящая при возникновении определенного события.

Механизм прерываний позволяет координировать параллельное функционирование отдельных устройств вычислительной системы и реагировать на особые состояния, возникающие при работе процессора. Идея прерывания была предложена в середине 50-х годов, основная цель введения механизма прерываний – реализация асинхронного (независимого для разных компонентов) режима работы и распараллеливание работы отдельных устройств.

Суть механизма прерываний заключается в следующем.

Во время выполнения ЭВМ текущей программы внутри машины и в связанной с ней внешней среде (например, в технологическом процессе, управляемом ЭВМ) могут возникать события, требующие немедленной реакции на них со стороны машины. Реакция состоит в том, что машина прерывает обработку текущей программы и переходит к выполнению некоторой другой программы, специально предназначенной для данного события. По завершении этой программы ЭВМ возвращается к выполнению прерванной программы.

Главные функции механизма прерываний:

* распознавание и классификация прерываний
* передача управления обработчику прерываний
* корректное возвращение к прерванной программе.

***Схема обработки прерываний:***



*Обозначения:*

**!** ­ Прерывание (сигнал – установление факта прерывания)

**1 ­** Идентификация прерывания

**2** ­ Отключение всех других прерываний

**3** ­ Смена контекста\_1 (сохранение состояния прерванного процесса из системных регистров, загрузка в системные регистры контекста соответствующего обработчика прерываний)

**4** ­ Обработка прерывания, включающая определение программы Q, которую следует запустить;

**5** ­ Смена контекста\_2 (загрузка в системные регистры контекста определенной на предыдущем шаге программы Q)

**6** ­ Установка прежнего режима системы прерываний.

Замечания:

1. Передача управления осуществляется аппаратно.
2. Программой Q может быть и прерванная программа Р.
3. Для корректной обработки прерывания очень важно запретить любые прерывания до окончания начатой обработки, поскольку иначе может возникать бесконечная последовательность вложенных прерываний. По окончанию обработки прерывания необходимо восстановить прежний режим системы прерываний, т.е. разрешить их.

Главное место в процедуре перехода к прерывающей программе занимают передача из соответствующего регистра (регистров) процессора в память (в частности, в стек) на сохранение текущего вектора состояния прерываемой программы (чтобы можно было вернуться к ее исполнению) и загрузка в регистр (регистры) процессора вектора прерывания прерывающей программы, к которой при этом переходит управление процессором.

*Программные прерывания* (по внутрипрограммной команде) являются синхронными операциями. Любая программа может инициировать синхронное программное прерывание путем выполнения программы INT. MS-DOS использует для взаимодействия с программами прерывания с номерами 20Н – 1FH, программы BIOS и прикладные программы использует другие номера. При этом распределение номеров условно и аппаратно не закреплено.