МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ

«БРАТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Кафедра управления в технических системах

УТВЕРЖДАЮ:
Проректор по учебной работе
_____ Е.И. Луковникова
«___» мая 2019 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Б1.В.17

НАПРАВЛЕНИЕ ПОДГОТОВКИ

27.03.04 Управление в технических системах

ПРОФИЛЬ ПОДГОТОВКИ

Управление и информатика в технических системах

Программа академического бакалавриата

Квалификация (степень) выпускника: бакалавр

Программа составлена в соответствии с федеральным государственным образовательным стандартом высшего образования по направлению подготовки 27.03.04 Управление в технических системах от 20.10.2015 г № 1171 и учебным планом ФГБОУ ВО «БрГУ» от 01.04.2019 г № 196 для заочной формы обучения набора 2019 года

СОДЕРЖАНИЕ ПРОГРАММЫ **Стр.**

1. ПЕРЕЧЕНЬ ПЛАНИРУЕМЫХ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ, СООТНЕСЕННЫХ С ПЛАНИРУЕМЫМИ РЕЗУЛЬТАТАМИ ОСВОЕНИЯ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Вид деятельности выпускника

Дисциплина охватывает круг вопросов, относящихся к научно-исследовательскому виду профессиональной деятельности выпускника в соответствии с компетенциями и видами деятельности, указанными в учебном плане.

Цель дисциплины

Приобретение умений и навыков исследования проблем в своей предметной области, выбора методов и средств их решения, анализа результатов теоретических и экспериментальных исследований.

Задачи дисциплины

Формирование способностей анализа результатов исследований, выбора методов и средств решения проблем в своей предметной области.

Код компетенции	Содержание компетенций	Перечень планируемых результатов обучения по дисциплине
1	2	3
ОПК-7	способность учитывать современные тенденции развития электроники, измерительной и вычислительной техники, информационных технологий в своей профессиональной деятельности	**знать:** – принципы организации операционных систем; – классификацию процессов и ресурсов **уметь:** - управлять файловой системой и управлением вводом/выводом **владеть:** – навыками настройки, программирования и управления системными ресурсами вычислительных машин
ПК-2	способность проводить вычислительные эксперименты с использованием стандартных программных средств с целью получения математических моделей	**знать:** – организацию и управление памятью **уметь:** -настраивать оборудование для выполнения заданных алгоритмов работы **владеть:** – навыками вычислительных экспериментов с использованием стандартных программных средств

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Дисциплина Б1.В.17 Системное программное обеспечение относится к вариативным дисциплинам.

Дисциплина Б1.В.17 Системное программное обеспечение базируется на знаниях, полученных при изучении дисциплины Б1.В.7 Информатика и Б1.Б.12 Информационные технологии.

Основываясь на изучении перечисленных дисциплин, Б1.В.17 Системное программное обеспечение представляет основу для преддипломной практики и подготовки к государственной итоговой аттестации.

Такое системное междисциплинарное изучение направлено на достижение требуемого ФГОС уровня подготовки по квалификации бакалавр.

3. РАСПРЕДЕЛЕНИЕ ОБЪЕМА ДИСЦИПЛИНЫ

3.1. Распределение объема дисциплины по формам обучения

Форма обучения	*Курс*	*Семестр*	*Трудоемкость дисциплины в часах*						*Курсовая работа (проект), контрольная работа, реферат, РГР*	*Вид промежуточной аттестации*
			Всего часов	*Аудиторных часов*	*Лекции*	*Лабораторные занятия*	*Практические занятия*	*Самостоятельная работа*		
1	2	3	4	5	6	7	8	9	10	11
Очная	4	7	108	51	17	34	-	57	-	зачет
Заочная	4	-	108	11	5	6	-	97	-	зачет
Заочная (ускоренное обучение)	2	-	108	8	5	3	-	100	-	зачет
Очно-заочная	-	-	-	-	-	-	-		-	-

3.2. Распределение объема дисциплины по видам учебных занятий и трудоемкости

Вид учебных занятий	*Трудоемкость (час.)*	*в т.ч. в интерактивной, активной, инновационной формах,(час.)*	*Распределение по семестрам, час*
			7
1	2	3	4
I. Контактная работа обучающихся с преподавателем (всего)	51	12	51
Лекции (Лк)	17	-	17
Лабораторные занятия (ЛЗ)	34	12	34
II.Самостоятельная работа обучающихся (СР)	57	-	57
Подготовка к лабораторным занятиям	47	-	47
Подготовка к зачету	10	-	10
III. Промежуточная аттестация зачет	+	-	+
Общая трудоемкость дисциплины 108 час.	108	-	108
3 зач. ед.	3	-	3

4. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

4.1. Распределение разделов дисциплины по видам учебных занятий - для очной формы обучения:

№ раздела и темы	*Наименование раздела и тема дисциплины*	*Трудоемкость, (час.)*	*Виды учебных занятий, включая самостоятельную работу обучающихся и трудоемкость; (час.)*		
			учебные занятия		*самостоятельная работа обучающихся**
			лекции	*лабораторные занятия*	
1	2	3	4	6	7
1.	**Функции и организация операционных систем (ОС). Обзор современных ОС.**	**7,5**	**1,5**	**-**	**6**
1.1.	Системное программное обеспечение.	2,5	0,5	-	2
1.2.	Операционная система	2,5	0,5	-	2
1.3.	Обзор современных ОС	2,5	0,5	-	2
2.	**Процессы. Операции над процессами. Процессы и нити. Идентификация и группирование процессов. Классификация процессов и ресурсов**	**13,5**	**1,5**	**7**	**5**
2.1.	Процессы и нити.	5,5	0,5	3	2
2.2.	Операции над процессами.	4,5	0,5	2	2
2.3.	Классификация процессов и ресурсов	3,5	0,5	2	1
3.	**Задачи синхронизации. Семафорная техника синхронизации**	**6,5**	**1,5**	-	**5**
3.1.	Задачи синхронизации	2,5	0,5	-	2
3.2.	Семафорная техника синхронизации	2,5	0,5	-	2
3.3.	Проблемы синхронизации	1,5	0,5		1
4.	**Тупики. Условия возникновения, предупреждение и обходы**	**6,5**	**1,5**	-	**5**
4.1.	Тупики	2,5	0,5	-	2
4.2.	Условия возникновения тупиков	2,5	0,5	-	2
4.3.	Предупреждение и обходы тупиков	1,5	0,5	-	1
5.	**Межпроцессорные коммуникации (сигнальный механизм, очереди сообщений, разделяемые сегменты памяти, сокеты)**	**6,5**	**1,5**	-	**5**
5.1.	Сигнальный механизм, очереди сообщений	2,5	0,5	-	2
5.2.	Разделяемые сегменты памяти	2,5	0,5	-	2
5.3.	Сокеты	1,5	0,5	-	1
6.	**Системные часы и таймеры. Планирование выполнения**	**13,5**	**1,5**	**7**	**5**

	процессов. Диспетчеризация процессов реального времени				
6.1.	Системные часы и таймеры	5,5	0,5	3	2
6.2.	Планирование выполнения процессов	4,5	0,5	2	2
6.3.	Диспетчеризация процессов реального времени	3,5	0,5	2	1
7.	**Организация и управление памятью**	**13,5**	**1,5**	**7**	**5**
7.1.	Страничное распределение	5,5	0,5	3	2
7.2.	Сегментное распределение	4,5	0,5	2	2
7.3.	Сегментно-страничное распределение	3,5	0,5	2	1
8.	**Файловая система. Управление вводом/выводом. Варианты структур ядра ОС**	**6,5**	**1,5**	-	**5**
8.1.	Файловая система	2,5	0,5	-	2
8.2.	Управление вводом/выводом.	2,5	0,5	-	2
8.3.	Варианты структур ядра ОС	1,5	0,5	-	1
9.	**Мультипроцессорные ОС. Сетевые ОС. Распределенные ОС.**	**5,5**	**1,5**	-	**4**
9.1.	Мультипроцессорные ОС	2,5	0,5	-	2
9.2.	Сетевые ОС	1,5	0,5	-	1
9.3.	Распределенные ОС	1,5	0,5	-	1
10.	**Вычислительный процесс. Обслуживание прерываний**	**9,5**	**1,5**	**4**	**4**
10.1.	Система прерывания	6,5	0,5	4	2
10.2.	Классификация типов прерывания	1,5	0,5	-	1
10.3.	Основные характеристики систем прерывания	1,5	0,5	-	1
11.	**Многозадачные и многопользовательские ОС. Распределение ресурсов в ОС.**	**8**	**1**	**3**	**4**
11.1.	Многозадачные и многопользовательские ОС	4,5	0,5	2	2
11.2.	Распределение ресурсов в ОС.	3,5	0,5	1	2
12.	**Системные программы: утилиты, макроассемблеры, компиляторы, интерпретаторы, отладчики. Сохранность и защита программных систем**	**11**	**1**	**6**	**4**
12.1.	Системные программы: утилиты, макроассемблеры, компиляторы, интерпретаторы, отладчики.	8,5	0,5	6	2
12.2.	Сохранность и защита программных систем	2,5	0,5	-	2
	ИТОГО	**108**	**17**	**34**	**57**

- для заочной формы обучения:

№ раздела и темы	***Наименование раздела и тема дисциплины***	***Трудоемкость, (час.)***	***Виды учебных занятий, включая самостоятельную работу обучающихся и трудоемкость; (час.)***		
			учебные занятия		***самостоя***
			лекции	***лаборатор-***	***стоя-***

				ные занятия	***тельная работа обучающихся****
1	**2**	**3**	**4**	**6**	**7**
1.	**Функции и организация операционных систем (ОС). Обзор современных ОС.**	**8,5**	**0,5**	**-**	**8**
1.1.	Системное программное обеспечение.	3,2	0,2	-	3
1.2.	Операционная система	3,2	0,2	-	3
1.3.	Обзор современных ОС	2,1	0,1	-	2
2.	**Процессы. Операции над процессами. Процессы и нити. Идентификация и группирование процессов. Классификация процессов и ресурсов**	**9,5**	**0,5**	**1**	**8**
2.1.	Процессы и нити.	4,2	0,2	1	3
2.2.	Операции над процессами.	3,2	0,2	-	3
2.3.	Классификация процессов и ресурсов	2,1	0,1	-	2
3.	**Задачи синхронизации. Семафорная техника синхронизации**	**8,4**	**0,4**	-	**8**
3.1.	Задачи синхронизации	3,2	0,2	-	3
3.2.	Семафорная техника синхронизации	3,1	0,1	-	3
3.3.	Проблемы синхронизации	2,1	0,1	-	2
4.	**Тупики. Условия возникновения, предупреждение и обходы**	**8,4**	**0,4**	-	**8**
4.1.	Тупики	3,2	0,2	-	3
4.2.	Условия возникновения тупиков	3,1	0,1	-	3
4.3.	Предупреждение и обходы тупиков	2,1	0,1	-	2
5.	**Межпроцессорные коммуникации (сигнальный механизм, очереди сообщений, разделяемые сегменты памяти, сокеты)**	**8,4**	**0,4**	-	**8**
5.1.	Сигнальный механизм, очереди сообщений	3,2	0,2	-	3
5.2.	Разделяемые сегменты памяти	3,1	0,1	-	3
5.3.	Сокеты	2,1	0,1	-	2
6.	**Системные часы и таймеры. Планирование выполнения процессов. Диспетчеризация процессов реального времени**	**9,4**	**0,4**	**1**	**8**
6.1.	Системные часы и таймеры	4,2	0,2	1	3
6.2.	Планирование выполнения процессов	3,1	0,1	-	3
6.3.	Диспетчеризация процессов реального времени	2,1	0,1	-	2
7.	**Организация и управление памятью**	**9,4**	**0,4**	**1**	**8**
7.1.	Страничное распределение	4,2	0,2	1	3
7.2.	Сегментное распределение	3,1	0,1	-	3

7.3.	Сегментно-страничное распределение	2,1	0,1	-	2
8.	**Файловая система. Управление вводом/выводом. Варианты структур ядра ОС**	**8,4**	**0,4**		**8**
8.1.	Файловая система	3,2	0,2	-	3
8.2.	Управление вводом/выводом.	3,1	0,1	-	3
8.3.	Варианты структур ядра ОС	2,1	0,1	-	2
9.	**Мультипроцессорные ОС. Сетевые ОС. Распределенные ОС.**	**8,4**	**0,4**		**8**
9.1.	Мультипроцессорные ОС	3,2	0,2	-	3
9.2.	Сетевые ОС	3,1	0,1	-	3
9.3.	Распределенные ОС	2,1	0,1	-	2
10.	**Вычислительный процесс. Обслуживание прерываний**	**8,4**	**0,4**	**1**	**7**
10.1.	Система прерывания	4,2	0,2	1	3
10.2.	Классификация типов прерывания	2,1	0,1	-	2
10.3.	Основные характеристики систем прерывания	2,1	0,1	-	2
11.	**Многозадачные и многопользовательские ОС. Распределение ресурсов в ОС.**	**8,4**	**0,4**	**1**	**7**
11.1.	Многозадачные и многопользовательские ОС	5,2	0,2	1	4
11.2.	Распределение ресурсов в ОС.	3,2	0,2	-	3
12.	**Системные программы: утилиты, макроассемблеры, компиляторы, интерпретаторы, отладчики. Сохранность и защита программных систем**	**8,4**	**0,4**	**1**	**7**
12.1.	Системные программы: утилиты, макроассемблеры, компиляторы, интерпретаторы, отладчики.	5,2	0,2	1	4
12.2.	Сохранность и защита программных систем	3,2	0,2	-	3
	ИТОГО	**104**	**5**	**6**	**93**

- для заочной формы обучения (ускоренное обучение)**:**

№ раздела и темы	***Наименование раздела и тема дисциплины***	***Трудоемкость, (час.)***	***Виды учебных занятий, включая самостоятельную работу обучающихся и трудоемкость; (час.)***		
			учебные занятия		***самостоятельная работа обучающихся****
			лекции	***лабораторные занятия***	
1	**2**	**3**	**4**	**6**	**7**
1.	**Функции и организация операционных систем (ОС). Обзор современных ОС.**	**8,5**	**0,5**	**-**	**8**
1.1.	Системное программное обеспечение.	3,2	0,2	-	3
1.2.	Операционная система	3,2	0,2	-	3

1.3.	Обзор современных ОС	2,1	0,1	-	2
2.	**Процессы. Операции над процессами. Процессы и нити. Идентификация и группирование процессов. Классификация процессов и ресурсов**	**9**	**0,5**	**0,5**	**8**
2.1.	Процессы и нити.	3,7	0,2	0,5	3
2.2.	Операции над процессами.	3,2	0,2	-	3
2.3.	Классификация процессов и ресурсов	2,1	0,1	-	2
3.	**Задачи синхронизации. Семафорная техника синхронизации**	**8,4**	**0,4**	-	**8**
3.1.	Задачи синхронизации	3,2	0,2	-	3
3.2.	Семафорная техника синхронизации	3,1	0,1	-	3
3.3.	Проблемы синхронизации	2,1	0,1	-	2
4.	**Тупики. Условия возникновения, предупреждение и обходы**	**8,4**	**0,4**	-	**8**
4.1.	Тупики	3,2	0,2	-	3
4.2.	Условия возникновения тупиков	3,1	0,1	-	3
4.3.	Предупреждение и обходы тупиков	2,1	0,1	-	2
5.	**Межпроцессорные коммуникации (сигнальный механизм, очереди сообщений, разделяемые сегменты памяти, сокеты)**	**8,4**	**0,4**	-	**8**
5.1.	Сигнальный механизм, очереди сообщений	3,2	0,2	-	3
5.2.	Разделяемые сегменты памяти	3,1	0,1	-	3
5.3.	Сокеты	2,1	0,1	-	2
6.	**Системные часы и таймеры. Планирование выполнения процессов. Диспетчеризация процессов реального времени**	**8,9**	**0,4**	**0,5**	**8**
6.1.	Системные часы и таймеры	3,7	0,2	0,5	3
6.2.	Планирование выполнения процессов	3,1	0,1	-	3
6.3.	Диспетчеризация процессов реального времени	2,1	0,1	-	2
7.	**Организация и управление памятью**	**8,9**	**0,4**	**0,5**	**8**
7.1.	Страничное распределение	3,7	0,2	0,5	3
7.2.	Сегментное распределение	3,1	0,1	-	3
7.3.	Сегментно-страничное распределение	2,1	0,1	-	2
8.	**Файловая система. Управление вводом/выводом. Варианты структур ядра ОС**	**8,4**	**0,4**		**8**
8.1.	Файловая система	3,2	0,2	-	3
8.2.	Управление вводом/выводом.	3,1	0,1	-	3
8.3.	Варианты структур ядра ОС	2,1	0,1	-	2
9.	**Мультипроцессорные ОС. Сетевые ОС. Распределенные ОС.**	**8,4**	**0,4**		**8**

9.1.	Мультипроцессорные ОС	3,2	0,2	-	3
9.2.	Сетевые ОС	3,1	0,1	-	3
9.3.	Распределенные ОС	2,1	0,1	-	2
10.	**Вычислительный процесс. Обслуживание прерываний**	**8,9**	**0,4**	**0,5**	**8**
10.1.	Система прерывания	3,7	0,2	0,5	3
10.2.	Классификация типов прерывания	3,1	0,1	-	3
10.3.	Основные характеристики систем прерывания	2,1	0,1	-	2
11.	**Многозадачные и многопользовательские ОС. Распределение ресурсов в ОС.**	**8,9**	**0,4**	**0,5**	**8**
11.1.	Многозадачные и многопользовательские ОС	4,7	0,2	0,5	4
11.2.	Распределение ресурсов в ОС.	4,2	0,2	-	4
12.	**Системные программы: утилиты, макроассемблеры, компиляторы, интерпретаторы, отладчики. Сохранность и защита программных систем**	**8,9**	**0,4**	**0,5**	**8**
12.1.	Системные программы: утилиты, макроассемблеры, компиляторы, интерпретаторы, отладчики.	4,7	0,2	0,5	4
12.2.	Сохранность и защита программных систем	4,2	0,2	-	4
	ИТОГО	**104**	**5**	**3**	**96**

4.2. Содержание дисциплины, структурированное по разделам и темам

Тема 1. Функции и организация операционных систем (ОС). Обзор современных ОС.

Системное программное обеспечение.

Системное программное обеспечение — это комплекс программ, которые обеспечивают эффективное управление компонентами вычислительной системы, такими как процессор, оперативная память, каналы ввода-вывода, сетевое и коммуникационное оборудование и т.п. Системное программное обеспечение реализует связь аппаратного и программного обеспечения, выступая как "межслойный интерфейс" с одной стороны которого аппаратура, а с другой приложения пользователя. Кроме системного программного обеспечения принято выделять Прикладное программное обеспечение, которое призвано решать прикладные задачи пользователя.

Операционная система.

Операцио́нная систе́ма, ОС (англ. operating system) — базовый комплекс компьютерных программ, обеспечивающий управление аппаратными средствами компьютера, работу с файлами, ввод и вывод данных, а также выполнение прикладных программ и утилит.

При включении компьютера операционная система загружается в память раньше остальных программ и затем служит платформой и средой для их работы. Помимо вышеуказанных функций ОС может осуществлять и другие, например, предоставление пользовательского интерфейса, сетевое взаимодействие и т. п.

Обзор современных ОС.

Windows

Семейство операционных систем корпорации Microsoft, базирующихся на основе графического интерфейса пользователя. Появление их явилось решающим шагом в широком продвижении и развитии перспективных способов взаимодействия систем человек-машина и машина-машина, создания дружественной среды для взаимодействия как пользователя с компьютерными приложениями, так и аппаратных средств внутри вычислительного комплекса.

В настоящее время под управлением операционных систем семейства Windows работает около 90% персональных компьютеров.

Первая версия Windows 1.0 вышла 20 ноября 1985 года. Последняя версия Windows 7 была выпущена в 2009 году.

Пакет Microsoft Windows включает в себя стандартные приложения, такие как браузер (Internet Explorer), почтовый клиент (Outlook Express или Windows Mail), музыкальный и видео проигрыватель (Windows Media Player). С помощью технологий COM и OLE их компоненты могут быть использованы в приложениях сторонних производителей. Эти продукты бесплатны, и могут быть свободно скачаны с официального сайта Microsoft, однако для установки некоторых из них необходимо иметь лицензионную версию Microsoft Windows. Запуск этих программ под другими операционными системами возможен только с помощью эмуляторов среды Windows (Wine), хотя такое их использование нарушает пользовательское соглашение.

Unix

Группа переносимых, многозадачных и многопользовательских операционных систем.

Первая система UNIX была разработана в 1969 г. в подразделении Bell Labs компании AT&T. С тех пор было создано большое количество различных UNIX-систем. Юридически лишь некоторые из них имеют полное право называться «UNIX»; остальные же, хотя и используют сходные концепции и технологии, объединяются термином «UNIX-подобные» (англ. Unix-like). Для краткости в данной статье под UNIX-системами подразумеваются как истинные UNIX, так и UNIX-подобные ОС.

Некоторые отличительные признаки UNIX-систем включают в себя:

использование простых текстовых файлов для настройки и управления системой;

широкое применение утилит, запускаемых в командной строке;

взаимодействие с пользователем посредством виртуального устройства — терминала;

представление физических и виртуальных устройств и некоторых средств межпроцессового взаимодействия как файлов;

использование конвейеров из нескольких программ, каждая из которых выполняет одну задачу.

В настоящее время UNIX используются в основном на серверах, а также как встроенные системы для различного оборудования. На рынке ОС для рабочих станций и домашнего применения UNIXуступили другим операционным системам, таким как Microsoft Windows и Mac OS, так как существующие программные решения для Unix-систем не позволяют реализовать полноценные рабочие станции ни для офисного, ни для домашнего использования.

UNIX-системы имеют большую историческую важность, поскольку благодаря им распространились некоторые популярные сегодня концепции и подходы в области ОС и программного обеспечения. Также, в ходе разработки Unix-систем был создан язык Си.

Идеи, заложенные в основу UNIX, оказали огромное влияние на развитие компьютерных операционных систем. В настоящее время UNIX-системы признаны одними из самых исторически важных ОС.

MacOS

Семейство операционных систем с графическим интерфейсом. Вместе с Mac OS X вторая по популярности в мире операционная система. Разработана корпорацией Apple (ранее — Apple Computer) для своей линейки компьютеров Macintosh. Популяризация графического интерфейса пользователя в современных операционных системах часто считается заслугой Mac OS. Она была впервые представлена в 1984 году вместе с оригинальным Macintosh 128K.

Apple хотела, чтобы Макинтош представлялся как компьютер «для всех остальных» («for the rest of us»). Сам термин «Mac OS» в действительности не существовал до тех пор, пока не был официально использован в середине 1990-х годов. С тех пор термин применяется ко всем версиям операционных систем Макинтоша как удобный способ выделения их в контексте других операционных систем.

Ранние версии Mac OS были совместимы только с Макинтошами, основанными на процессорах Motorola 68k, следующие версии были совместимы с архитектурой PowerPC (PPC). С недавних пор Mac OS стала совместима с архитектурой Intel x86. Но политика фирмы Apple такова, что она разрешает устанавливать систему Mac OS только на компьютеры Apple.

Тема 2. Процессы. Операции над процессами. Процессы и нити. Идентификация и группирование процессов. Классификация процессов и ресурсов.

Процессы и нити.

Общее понятие процесса для ОС Windows как бы распадается на два понятия: собственно процесса и нити (thread; в некоторых книгах используется термин поток). При этом нить является единицей работы, она участвует в конкуренции за процессорное время, изменяет свое состояние и приоритет, как было описано выше для процесса. Что же касается процесса в Windows, то он может

состоять из нескольких нитей, использующих общую память, открытые файлы и другие ресурсы, принадлежащие процессу. В двух словах: процесс – владеет (памятью, файлами), нити – работают, при этом совместно используя ресурсы своего процесса. Правда, нить тоже кое-чем владеет: окнами, очередью сообщений, стеком.

Процесс создается при запуске программы (EXE-файла). Одновременно создается одна нить процесса (должен же кто-то работать!). Создание процесса выполняется с помощью API-функции CreateProcess. Основными параметрами при вызове этой функции являются следующие.

· Имя файла запускаемой программы.
· Командная строка, передаваемая процессу при запуске.
· Атрибуты защиты для создаваемых процесса и нити. И процесс, и нить являются объектами ядра Windows и в этом качестве могут быть защищены от несанкционированного доступа (например, от попыток других процессов вмешаться в работу данного процесса).
· Различные флаги, уточняющие режим создания процесса. Среди них следует отметить класс приоритета процесса, флаг отладочного режима (при этом система будет уведомлять процесс-родитель о действиях порожденного процесса), а также флаг создания приостановленного процесса, который не начнет работать, пока не будет вызвана функция возобновления работы.
· Блок среды процесса.
· Текущий каталог процесса.
· Параметры первого окна, которое будет открыто при запуске процесса.
· Адрес блока информации, через который функция возвращает родительскому процессу четыре числа: идентификатор созданного процесса, идентификатор нити, хэндл процесса и хэндл нити.

Если процесс успешно создан, функция CreateProcess возвращает ненулевое значение.

Операции над процессами.

Системы, управляющие процессами, должны иметь возможность выполнять над ними ряд операций.

Создание процесса.

Создание процесса включает присвоение имени процессу; включение его имени в список имен процессов; определение начального приоритета процесса; формирование блока управления процессом PCB; выделение процессу начальных ресурсов.

Процесс может породить новый процесс и в этом случае, первый будет называться родительским, а второй дочерним процессом, причем у одного родительского процесса может быть несколько дочерних, а у дочернего только один родительский. Таким образом, создается иерархическая структура процессов.

ОС UNIX, являясь в своей основе средством управления процессами, сама по себе может рассматриваться как система параллельных взаимодействий процессов с древовидной структурой. Общий прародитель всех процессов в ОС UNIX - процесс init, находится в вершине генеалогического дерева, этот процесс постоянно присутствует в системе, все другие процессы порождаются по унифицированной схеме с помощью системного вызова fork().

Каждому созданному процессу UNIX назначает уникальный идентификатор процесса - PID , который идентифицирует процесс для ОС. Кроме того, каждый процесс имеет еще PPID (parent process), который представляет собой не что иное как PID его родителя.

Используя в UNIX команду ps можно видеть идентификаторы текущих процессов в системе.

Уничтожение процесса.

При уничтожении процесса, ресурсы ему выделенные передаются системе, имя из любых списков и таблиц удаляется, а блок управления процессом освобождается.

Приостановка процесса.

Приостановленный процесс может продолжить свое выполнение тогда, когда его активизирует какой-либо другой процесс.

Возобновление процесса.

Операция подготовки процесса к повторному запуску с той точки , в которой он был приостановлен, называется - возобновлением.

Изменение приоритета процесса.

Эта операция означает модификацию значения приоритета процесса в PCB.

Кроме того, используются операции блокирования, пробуждения и запуска процесса.

Классификация процессов и ресурсов

Классификация процессов

1. По времени существования

- *процессы реального времени* – такой вид процесса, требующий такого планирования, при котором гарантировалось бы окончание процесса до наступления конкретного времени.
- *интерактивные* – время существования таких процессов должно быть не более интервала времени допустимой реакции ЭВМ на запросы пользователя.
- *пакетные* (остальные).

2. По генеалогическому признаку
   - *порождающие* – процесс, задающий требования для порождения других процессов.
   - *порожденные* – процессы, создаваемые по требованию других.
3. По результативности
   - *эквивалентные* – процессы, имеющие одинаковый конечный результат при использовании одних и тех же данных, с использованием одних и тех же или различных программ или процессоров.
   - *тождественные* – эквивалентные процессы. Обработка данных происходит по одной и той же программе, но трассы которых не совпадают.
   - *равные* – тождественные процессы, трассы которых совпадают.
   - *различные* – все остальные
4. По принадлежности к центральному процессору
   - *внутренние* – развивающиеся в процессоре.
   - *внешние* – развитие которых происходит под контролем или управлением ОС на другом процессоре.
5. По принадлежности к ОС
   - *системные* – при развитии которых исполняется программа из состава ОС
   - *пользовательские* – при развитии которых выполняется прикладная программа
6. По динамическому признаку – соотношение интервалов существования процессов
   - *последовательные* – интервалы которых не пересекаются во времени
   - *параллельные* – существуют одновременно на данном интервале времени
   - *комбинированные* (пересекающиеся)
7. По связанности процессов
   - *взаимосвязанные* – между процессами с помощью системы управления процессами поддерживаются связи какого-либо рода (функциональные, пространственно-временные, управляющие, информационные и т.д.)
   - *информационно независимые* – взаимосвязанные процессы, при развитии которых используется совместно некоторый ресурс, но информационно они не связаны
   - *взаимодействующие* – взаимосвязанные процессы с информационной связью
   - *конкурирующие* - взаимосвязанные процессы, имеющие связь по ресурсам.

Ресурсы

Ресурс вычислительной системы - средство вычислительной системы, которое может быть выделено процессу обработки данных на определенный интервал времени.

Основными ресурсами вычислительной системы являются: процессоры; области основной памяти; наборы данных; периферийные устройства; программы.

Классификация ресурсов

1. По реальности существования
   - *Физический* – это ресурс, который реально существует и при распределении обладает всеми физическими свойствами и характеристиками.
   - *Виртуальный* – мнимый ресурс, модель некоторого ресурса, которая реализуется в программно- аппаратной форме и имеет преимущество над физическим.
2. По возможности расширения свойств, т.е. по возможности построения виртуального ресурса
   - *Эластичный*, т.е. допускает виртуализацию
   - *Жесткий* (неэластичный), т.е. не допускает создание виртуального ресурса
3. По степени активности
   - *Активные*, при использовании они способны выполнять действия по отношению к другим ресурсам, которые приводят к изменению последних (например, центральный процессор).
   - *Пассивные*, над которыми можно производить дополнительные действия, которые приводят к их изменению (например, ОП).
4. По времени существования
   - *Постоянные*, т.е. существуют до рождения процесса, во время существования процесса и возможно будут существовать после процесса (например, ПЗУ).

- *Временные* – появляющиеся или уничтожающиеся в системе динамически в течение времени существования процесса. Создание или уничтожение может производиться самим процессом или другими процессами.

5. По степени важности
   - *Главные* – без выделения этих ресурсов процесс принципиально существовать не может (например, центральный процессор и ОП)
   - *Второстепенные* - ресурсы, в отсутствие которых возможно некое альтернативное развитие процесса.
6. По функциональной избыточности:
   - *Дорогие*
   - *Дешевые*
7. По структуре
   - *Простые* – не содержат составные части
   - *Составные*
8. По восстанавливаемости
   - *Воспроизводимые ресурсы* – ресурсы, при распределении которых допускается многократное выполнение следующей последовательности:

     Запрос ⇒ Выделение⇒ Использование ⇒ Освобождение
   - В отношении определенной категории ресурсов многократное применение последовательности З-В-И-О невозможно, поскольку на каком-либо цикле работы с ними может наступить ситуация исчерпания ресурса, т.е. обрыв последовательности на шаге ИСПОЛЬЗОВАНИЕ, после чего такой ресурс изымается из использования. Ресурсы с таким свойством называются *потребляемыми*.
9. По характеру использования
   - *Последовательно-используемый* – это ресурс, который одновременно может использоваться только одним процессом.
   - *Параллельно-используемый* - это ресурс, который одновременно используется несколькими одновременно работающими процессами. Например, массив данных находится в некоторой области ОП и допускает чтение этих данных из нее.

Тема 3. Задачи синхронизации. Семафорная техника синхронизации

Задачи синхронизации

Известны типичные задачи синхронизации, к которым могут быть сведены практически все известные способы упорядочения работ во времени. Эти задачи, как правило, решаются при реализации второго способа распараллеливания.

1. Задача взаимного исключения. Имеется несколько процессов, программы которых содержат критические интервалы, где процессы обращаются к одному разделяемому критическому ресурсу (например, к базе знаний). Требуется исключить одновременное вхождение в такие критические интервалы более чем одного процесса.

Требования к решению этой задачи:

- задержка любого процесса вне его критического интервала не должна влиять на развитие других процессов;
- решение должно быть симметрично относительно процессов;
- решение не должно допускать тупиков.

2. Задача "поставщики — потребители". Имеется ограниченный буфер на несколько порций информации. Он является критическим ресурсом для процессов двух типов:

- процессы "поставщики", получая доступ к ресурсу, помещают на свободное место в буфере несколько или одну порцию информации;
- процессы "потребители", получая доступ к ресурсу, считывают из него порции информации.

Требуется исключить одновременный доступ к ресурсу любых двух процессов. При опустошении буфера следует задерживать процессы "потребители", при полном заполнении буфера — процессы "поставщики".

Эта задача возникает, например, при обмене с внешними устройствами и заключается в программной имитации кольцевого или бесконечного буфера.

3. Задача "читатели — писатели".

Имеется разделяемый ресурс — область памяти, к которой требуется доступ процессам двух типов:

Процессы первого типа — "ЧИТАТЕЛИ" — могут получать доступ к разделяемому ресурсу одновременно. Они считывают информацию.

Процессы второго типа — "ПИСАТЕЛИ" — взаимно исключают и друг друга, и "читателей". Они записывают в разделяемую область памяти данные.

Задача известна в двух вариантах:

1. "Читатели", изъявившие желание получить доступ к ресурсу, должны получить его как можно быстрее; это — первая задача ЧП.
2. "Читатели", изъявившие желание получить доступ к ресурсу, должны получить его как можно быстрее, если отсутствуют запросы от "писателей". "Писатель", требующий доступ к ресурсу, должен получить его как можно быстрее, но после обслуживания "читателей", подошедших к ресурсу до первого "писателя". Это — вторая задача ЧП.

Приведем возможное решение задач с помощью комбинированного семафора.

Считаем, что процедура ОТКРЫТЬ ПО СЧИТЫВАНИЮ выполняется подобно процедуре ПРОПУСТИТЬ, изменяя только значение семафора-счетчика. Процедура ОТКРЫТЬ ПО ЗАПИСИ выполняется подобно процедуре ОТКРЫТЬ, "открывая" семафор и обеспечивая запуск "задержанных" процессов с процедур ЗАКРЫТЬ ПО ЗАПИСИ или ЖДАТЬ ПО ЗАПИСИ, при выполнении которых произошло прерывание.

4. Задача "обедающие философы". За круглым столом сидят k философов, которые проводят время, чередуя философские размышления с потреблением пищи. Перед каждым — тарелка спагетти, между тарелками — по одной вилке. Для еды каждому философу требуются две вилки. Использовать можно только вилки, лежащие рядом с тарелками. Так как переходы от размышления к принятию пищи производятся в непредсказуемые моменты времени, то возможны конфликты и требуется синхронизация процессов.

Представим следующую модель, требующую решение данной задачи, — модель оперативного обмена между процессорами векторной ВС или строк (столбцов) матричной ВС (рис. 1).

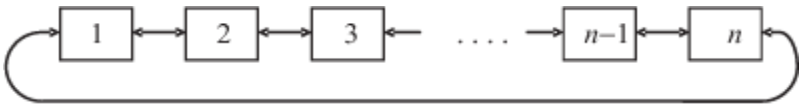


Рис. 1. Связь по схеме "обедающие философы"

Например, после счета очередного элемента в узле сетки результаты должны быть переданы соседним процессорным элементам для использования в следующей итерации. Очевидна возможность конфликтов при попытке одновременной встречной передачи.

Пусть с i-м процессором для передачи влево связан "левый" семафор C_i, для передачи вправо — "правый" семафор C_{i+1} (или наоборот). Пусть каждый процессор, нуждающийся в передаче двум соседям, пытается сначала закрыть свой "правый" (аналогично, "левый") семафор. Затем, если это не успел сделать левый сосед, он попытается закрыть "левый" семафор и произвести передачу. Тогда возможен общий тупик, если все процессоры одновременно закроют все семафоры. (При одновременном пересчете значений функции в узлах сетки вероятность такой ситуации высока.)

Разрешим четным процессорам сначала закрывать "левые" ("правые") семафоры, а нечетным — "правые" ("левые"). Тогда схемы программ для них будут выглядеть следующим образом:

5.Задача обновления данных. Предполагает запрет на использование обновляемых данных. Например, процессор обновляет запись в базе данных. В простейшем случае может быть использован признак, по которому запрещается обращение к данной записи, пока она не будет обновлена.

Семафорная техника синхронизации

В 1965 году Деккерер ввёл понятие «семафорная техника».

Все виды решения задач синхронизации базируются на двух примитивах**: P** и **V.**

Примитив **P** ставится впереди критической области, а **V** – в конце критической области. В каждом процессе они есть. Эти примитивы неделимы и взаимно исключают друг друга. Они основаны на использовании семафоров.

Семафор **–** это системная переменная специального типа, над которой можно производить две операции: открытия и закрытия. Семафорный механизм работает по двухэтапной схеме и использует механизм пассивного ожидания. В состав семафорного механизма включены средства формирования и обслуживания очередей процессов, которым не удаётся успешно выполнить операцию закрытия семафора**.**

Параметры семафора:

*1)*начальное значение семафора;

2)диапазон значений;3)логика действий над семафорами (задаётся процедурами обработки);4)количество семафоров, доступных для обработки отдельных примитивов**.**

Алгоритм P(S)

{Закрыть ресурс

S – семафор; S=0 – семафор закрыт; S=1 – открыт}

S:=S-1;

if S<0 then <остановить процесс и поместить его в очередь ожидания к семафору S>

else <продолжить процесс>

Алгоритм V(S)

{открыть доступ к ресурсу}

If S<=0 then <поместить один из ожидающих процессов из очереди к семафору S в очередь готовности>;

S:=S+1;

Проблемы синхронизации

Синхронизация в компьютерных системах — это координация работы процессов таким образом, чтобы последовательность их операций была предсказуемой. Как правило, синхронизация необходима при совместном доступе к разделяемым ресурсам. Критическая секция — часть программы, в которой есть обращение к совместно используемым данным. При нахождении в критической секции двух (или более) процессов, возникает состояние гонки/состязания за ресурсы. Состояние гонки — это состояние системы, при котором результат выполнения операций зависит от того, в какой последовательности будут выполняться отдельные процессы в этой системе, но управлять этой последовательностью нет возможности. Иными словами, это противоположность правильной синхронизации. Для избежания гонок необходимо выполнение таких условий: · Взаимного исключения: два процесса не должны одновременно находиться в одной критической области · Прогресса: процесс, находящийся вне критической области, не может блокировать другие процессы · Ограниченного ожидания: невозможна ситуация, в которой процесс вечно ждет попадания в критическую область · Также в программе, в общем случае, не должно быть предположений о скорости или количестве процессоров Пример условий гонок: i++ movl i, $eax // i - метка адреса переменной i в памяти incl $eax movl $eax, i Поскольку процессор не может модифицировать значения в памяти непосредственно, для этого ему нужно загрузить значение в регистр, изменить его, а затем снова выгрузить в память. Если в процессе выполнения этих трех инструкций процесс будет прерван, может возникнуть непредсказуемый результат, т.е. имеет место условие гонки.

Тема 4. Тупики. Условия возникновения, предупреждение и обходы.

Тупики

Тупик (deadlock) - это такая ситуация в мультипрограммной системе, когда процесс ожидает некоторого события, которое никогда не произойдет. Системная тупиковая ситуация, или “зависание” системы - это ситуация, когда один или более процессов оказываются в состоянии тупика.

В ОС тупики в большинстве случаев возникают при конкуренции процессов за выделение ресурсов последовательного доступа, которые в каждый момент времени отводятся только одному пользователю.

Условия возникновения тупиков

Условия возникновения тупиков были сформулированы Коффманом, Элфиком и Шошани в 1970 г.:

4. Условие взаимоисключения (Mutual exclusion). Одновременно использовать ресурс может только один процесс.
5. Условие ожидания ресурсов (Hold and wait). Процессы удерживают ресурсы, уже выделенные им, и могут запрашивать другие ресурсы.
6. Условие неперераспределяемости (No preemtion). Ресурс, выделенный ранее, не может быть принудительно забран у процесса. Освобождены они могут быть только процессом, который их удерживает.
7. Условие кругового ожидания (Circular wait). Существует кольцевая цепь процессов, в которой каждый процесс ждет доступа к ресурсу, удерживаемому другим процессом цепи.

Для образования тупика необходимым и достаточным является выполнение *всех четырех* условий.

Предупреждение и обходы тупиков

Цель предотвращения тупиков – обеспечить условия, исключающие возможность возникновения тупиковых ситуаций. Большинство методов связано с предотвращением одного из условий возникновения взаимоблокировки.

Система, предоставляя ресурс в распоряжение процесса, должна принять решение, безопасно это или нет. Возникает вопрос: есть ли такой алгоритм, который помогает всегда избегать тупиков и делать правильный выбор. Ответ – да, мы можем избегать тупиков, но только если определенная информация известна заранее.

1) Способы предотвращения тупиков путем тщательного распределения ресурсов. Алгоритм банкира

Можно избежать взаимоблокировки, если распределять ресурсы, придерживаясь определенных правил. Среди такого рода алгоритмов наиболее известен алгоритм банкира, предложенный Дейкстрой, который базируется на так называемых *безопасных* или *надежных* состояниях (safe state). Безопасное состояние – это такое состояние, для которого имеется по крайней мере одна последовательность событий, которая не приведет к взаимоблокировке. Модель алгоритма основана на действиях банкира, который, имея в наличии капитал, выдает кредиты.

Суть алгоритма состоит в следующем.

- Предположим, что у системы в наличии n устройств, например лент.
- ОС принимает запрос от пользовательского процесса, если его максимальная потребность не превышает n.
- Пользователь гарантирует, что если ОС в состоянии удовлетворить его запрос, то все устройства будут возвращены системе в течение конечного времени.
- Текущее состояние системы называется *надежным*, если ОС может обеспечить всем процессам их выполнение в течение конечного времени.
- В соответствии с алгоритмом банкира выделение устройств возможно, только если состояние системы остается надежным.

2) Предотвращение тупиков за счет нарушения условий возникновения тупиков

В отсутствие информации о будущих запросах единственный способ избежать взаимоблокировки – добиться невыполнения хотя бы одного из условий раздела «Условия возникновения тупиков».

a) Нарушение условия взаимоисключения

В общем случае избежать взаимоисключений невозможно. Доступ к некоторым ресурсам должен быть исключительным. Тем не менее некоторые устройства удается обобществить. В качестве примера рассмотрим принтер. Известно, что пытаться осуществлять вывод на принтер могут несколько процессов. Во избежание хаоса организуют промежуточное формирование всех выходных данных процесса на диске, то есть разделяемом устройстве. Лишь один системный процесс, называемый сервисом или демоном принтера, отвечающий за вывод документов на печать по мере освобождения принтера, реально с ним взаимодействует. Эта схема называется спулингом (spooling). Таким образом, принтер становится разделяемым устройством, и тупик для него устранен.

К сожалению, не для всех устройств и не для всех данных можно организовать спулинг.

b) Нарушение условия ожидания дополнительных ресурсов

Условия ожидания ресурсов можно избежать, потребовав выполнения стратегии двухфазного захвата:

- В первой фазе процесс должен запрашивать все необходимые ему ресурсы сразу. До тех пор пока они не предоставлены, процесс не может продолжать выполнение.
- Если в первой фазе некоторые ресурсы, которые были нужны данному процессору, уже заняты другими процессами, он освобождает все ресурсы, которые были ему выделены, и пытается повторить первую фазу.

В известном смысле этот подход напоминает требование захвата всех ресурсов заранее. Естественно, что только специально организованные программы могут быть приостановлены в течение первой фазы и рестартованы впоследствии.

Таким образом, один из способов – заставить все процессы затребовать нужные им ресурсы перед выполнением («все или ничего»). Если система в состоянии выделить процессу все необходимое, он может работать до завершения. Если хотя бы один из ресурсов занят, процесс будет ждать.

c) Нарушение принципа отсутствия перераспределения

Если бы можно было отбирать ресурсы у удерживающих их процессов до завершения этих процессов, то удалось бы добиться невыполнения третьего условия возникновения тупиков. Перечислим минусы данного подхода.

Во-первых, отбирать у процессов можно только те ресурсы, состояние которых легко сохранить, а позже восстановить, например состояние процессора. Во-вторых, если процесс в течение не-

которого времени использует определенные ресурсы, а затем освобождает эти ресурсы, он может потерять результаты работы, проделанной до настоящего момента. Наконец, следствием данной схемы может быть дискриминация отдельных процессов, у которых постоянно отбирают ресурсы.

Весь вопрос в цене подобного решения, которая может быть слишком высокой, если необходимость отбирать ресурсы возникает часто.

d) Нарушение условия кругового ожидания

Трудно предложить разумную стратегию, чтобы избежать последнего условия из раздела «Условия возникновения тупиков» – циклического ожидания.

Один из способов – упорядочить ресурсы. Например, можно присвоить всем ресурсам уникальные номера и потребовать, чтобы процессы запрашивали ресурсы в порядке их возрастания. Тогда круговое ожидание возникнуть не может. После последнего запроса и освобождения всех ресурсов можно разрешить процессу опять осуществить первый запрос. Очевидно, что практически невозможно найти порядок, который удовлетворит всех.

Тема 5. Межпроцессорные коммуникации (сигнальный механизм, очереди сообщений, разделяемые сегменты памяти, сокеты).

Сигнальный механизм, очереди сообщений

В ОС Unix присутствует т.н. аппарат сигналов, позволяющий одним процессам оказывать воздействия на другие процессы. Сигналы могут рассматриваться как средство уведомления процесса о некотором наступившем в системе событии. В некотором смысле аппарат сигналов имеет аналогию с аппаратом прерываний, поскольку последний есть также уведомление системы о том, что в ней произошло некоторое событие. Прерывание вызывает определенную детерминированную последовательность действий системы, точно так же приход сигнала в процесс вызывает в нем определенную последовательность действий. Инициатором отправки сигнала процессу может быть как процесс или ОС. Для иллюстрации приведем следующий пример. Пускай в ходе выполнения некоторого процесса произошло деление на ноль, вследствие чего в системе происходит прерывание, управление передается операционной системе. ОС «видит», что это прерывание «деление на ноль», и отправляет сигнал процессу, в теле которого произошла данная ошибка. Дальше процесс реагирует на получение сигнала, но об этом чуть позже. 116 Инициатором посылки сигнала может выступать другой процесс. В качестве примера можно привести следующую ситуацию. Пользователь ОС Unix запустил некоторый процесс, который в некоторый момент времени зацикливается. Чтобы снять этот процесс со счета, пользователь может послать ему сигнал об уничтожении (например, нажав на клавиатуре комбинацию клавиш Ctrl+C, а это есть команда интерпретатору команд послать код сигнала SIGINT). В данном случае процесс интерпретатора команд пошлет сигнал пользовательскому процессу. Аппарат сигналов является механизмом асинхронного взаимодействия, момент прихода сигнала процессу заранее неизвестен. Так же, как и аппарат прерываний, имеющий фиксированное количество различных прерываний, Unix-системы имеют фиксированный набор сигналов. Перечень сигналов, реализованных в конкретной операционной системе, обычно находится в файле signal.h. В этом файле перечисляется набор пар «имя сигнала — его целочисленное значение». При получении процессом сигнала возможны 3 типа реакции на него. Во-первых, это обработка сигнала по умолчанию. В подавляющем большинстве случаев обработка сигнала по умолчанию означает завершение процесса. В этом случае системным кодом завершения процесса становится номер пришедшего сигнала. Во-вторых, процесс может перехватывать обработку пришедшего сигнала. Если процесс получает сигнал, то вызывается функция, принадлежащая телу процесса, которая была специальным образом зарегистрирована в системе как обработчик сигнала. Следует отметить, что часть реализованных в ОС сигналов можно перехватывать, а часть сигналов перехватывать нельзя. Примером неперехватываемого сигнала может служить сигнал SIGKILL (код 9), предназначенный для безусловного уничтожения процесса. А упомянутый выше сигнал SIGINT (код 2) перехватить можно. В-третьих, сигналы можно игнорировать, т.е. приход некоторых сигналов процесс может проигнорировать. Как и в случае с перехватываемыми сигналами, часть сигналов можно игнорировать (например, SIGINT), а часть — нет (например, SIGKILL). Для отправки сигнала в ОС Unix имеется системный вызов kill().

Очереди сообщений представляют собой связный список в адресном пространстве ядра. Сообщения могут посылаться в очередь по порядку и доставаться из очереди несколькими разными путями. Каждая очередь сообщений однозначно определена идентификатором IPC. Очереди сообщений как средство межпроцессной связи позволяют процессам взаимодействовать, обмениваясь данными. Данные передаются между процессами дискретными порциями, называемымисообщениями. Процессы, использующие этот тип межпроцессной связи, могут выполнять две операции:послать или принять сообщение.

Разделяемые сегменты памяти

Разделяемая память может быть наилучшим образом описана как отображение участка (сегмента) памяти, которая будет разделена между более чем одним процессом. Это гораздо более быстрая форма IPC, потому что здесь нет никакого посредничества (т.е. каналов, очередей сообщений и т.п.). Вместо этого, информация отображается непосредственно из сегмента памяти в адресное пространство вызывающего процесса. Сегмент может быть создан одним процессом и впоследствии использован для чтения/записи любым количеством процессов.

Сокеты

Сокеты обеспечивают двухстороннюю связь типа точка-точка между двумя процессами. Они являются основными компонентами межсистемной и межпроцессной связи. Каждый сокет представляет собой конечную точку связи, с которой может быть совмещено некоторое имя. Он имеет определенный тип, и один процесс или несколько, связанных с ним процессов.

Основные типы сокетов

Поточный - обеспечивает двухсторонний, последовательный, надежный, и недублированный поток данных без определенных границ. Тип сокета - SOCK_STREAM, в домене Интернета он использует протокол TCP.

Датаграммный- поддерживает двухсторонний поток сообщений. Приложение, использующее такие сокеты, может получать сообщения в порядке, отличном от последовательности, в которой эти сообщения посылались. Тип сокета - SOCK_DGRAM, в домене Интернета он использует протокол UDP.

Сокет последовательных пакетов - обеспечивает двухсторонний, последовательный, надежный обмен датаграммами фиксированной максимальной длины. Тип сокета - SOCK_SEQPACKET. Для этого типа сокета не существует специального протокола.

Простой сокет - обеспечивает доступ к основным протоколам связи.

Тема 6. Системные часы и таймеры. Планирование выполнения процессов. Диспетчеризация процессов реального времени.

Системные часы и таймеры

Часами называется программный или аппаратный объект, предназначенный для измерения видимого или истинного хода времени.

Показания часов можно опросить и установить (в допустимых для часов пределах).

Разрешающей способностью часов называется минимальный промежуток времени, который может быть этими часами измерен.

Под скачком часов понимается разность между двумя последовательными, различными с точки зрения приложения (использующего стандартные средства) показаниями часов.

Часы называются монотонными, если их нельзя установить стандартными средствами и они не могут иметь отрицательных скачков.

Такт часов - это зависящие от реализации промежутки времени, на которые дробится каждая секунда.

Реальным (или астрономическим) называется время, измеренное по системным часам безотносительно к тому, какой процесс (поток управления) выполняется.

Под временем выполнения (процессорным временем) понимается время, затрачиваемое на выполнение процесса (потока управления), включая работающие от его имени системные сервисы.

Время выполнения конкретного процесса или потока управления измеряется часами процессорного времени.

Под мониторингом времени выполнения понимается оперативное отслеживание процессорного времени, затрачиваемого процессом (потоком управления).

Виртуальное время процесса - время, измеряемое системными часами, пока процесс выполняется.

Таймер - это механизм, способный известить процесс (поток управления) об истечении заданного временного промежутка (подобный таймер называется интервальным) или о достижении (или превышении) часами заданных показаний (абсолютный таймер). Соответствующее событие называется срабатыванием таймера.

Таймером процессорного времени называется таймер, ассоциированный с часами процессорного времени.

Взвести (зарядить) - значит запустить таймер, измеряющий ход времени и позволяющий уведомить процесс о наступлении заданного момента.

Таймер снимается со взвода (разряжается), когда он перестает измерять ход времени, отключая тем самым будущие уведомления процесса до нового взведения таймера.

Таймеры можно подразделить на одноразовые и периодические.

Одноразовые таймеры снимаются со взвода после первого (и единственного) срабатывания.

Периодические таймеры после очередного срабатывания запускаются вновь в соответствии с однажды заданной спецификацией следующего срабатывания.

Под избыточным срабатыванием таймера понимается ситуация, когда таймер срабатывает до того, как процесс обработал сигнал о предыдущем срабатывании.

Планирование выполнения процессов

В разных ОС процессы реализуются по-разному. Эти различия заключаются в том, какими структурами данных представлены процессы, как они именуются, какими способами защищены друг от друга и какие отношения существуют между ними. Процессы Windows NT имеют следующие характерные свойства:

- Процессы Windows NT реализованы в форме объектов, и доступ к ним осуществляется посредством службы объектов.
- Процесс Windows NT имеет многонитевую организацию.
- Как объекты-процессы, так и объекты-нити имеют встроенные средства синхронизации.
- Менеджер процессов Windows NT не поддерживает между процессами отношений типа "родитель-потомок".

В любой системе понятие "процесс" включает следующее:

- исполняемый код,
- собственное адресное пространство, которое представляет собой совокупность виртуальных адресов, которые может использовать процесс,
- ресурсы системы, такие как файлы, семафоры и т.п., которые назначены процессу операционной системой.
- хотя бы одну выполняемую нить.

Адресное пространство каждого процесса защищено от вмешательства в него любого другого процесса. Это обеспечивается механизмами виртуальной памяти. Операционная система, конечно, тоже защищена от прикладных процессов. Чтобы выполнить какую-либо процедуру ОС или прочитать что-либо из ее области памяти, нить должна выполняться в режиме ядра. Пользовательские процессы получают доступ к функциям ядра посредством системных вызовов. В пользовательском режиме выполняются не только прикладные программы, но и защищенные подсистемы Windows NT.

Менеджер процессов определяет атрибуты, хранимые в теле объекта-процесса, а также обеспечивает системный сервис, который восстанавливает и изменяет эти атрибуты.

В число атрибутов тела объекта-процесса входят:

- Идентификатор процесса - уникальное значение, которое идентифицирует процесс в рамках операционной системы.
- Токен доступа - исполняемый объект, содержащий информацию о безопасности.
- Базовый приоритет - основа для исполнительного приоритета нитей процесса.–
- Процессорная совместимость - набор процессоров, на которых могут выполняться нити процесса.

Предельные значения квот - максимальное количество страничной и нестраничной системной памяти, дискового пространства, предназначенного для выгрузки страниц, процессорного времени - которые могут быть использованы процессами пользователя.

Время исполнения - общее количество времени, в течение которого выполняются все нити процесса.

Объект-нить имеет следующие атрибуты тела:

Идентификатор клиента - уникальное значение, которое идентифицирует нить при ее обращении к серверу.–

Контекст нити - информация, которая необходима ОС для того, чтобы продолжить выполнение прерванной нити. Контекст нити содержит текущее состояние регистров, стеков и индивидуальной области памяти, которая используется подсистемами и библиотеками.–

Динамический приоритет - значение приоритета нити в данный момент.–

Базовый приоритет - нижний предел динамического приоритета нити.–

Процессорная совместимость нитей - перечень типов процессоров, на которых может выполняться нить.–

Время выполнения нити - суммарное время выполнения нити в пользовательском режиме и в режиме ядра, накопленное за период существования нити.–

Состояние предупреждения - флаг, который показывает, что нить должна выполнять вызов асинхронной процедуры.–

Счетчик приостановок - текущее количество приостановок выполнения нити.–

В ОС Windows NT нить в ходе своего существования может иметь одно из шести состояний (рисунок 1.3). Жизненный цикл нити начинается в тот момент, когда программа создает новую нить. Запрос передается NT executive, менеджер процессов выделяет память для объекта-нити и обращается к ядру, чтобы инициализировать объект-нить ядра. После инициализации нить проходит через следующие состояния:

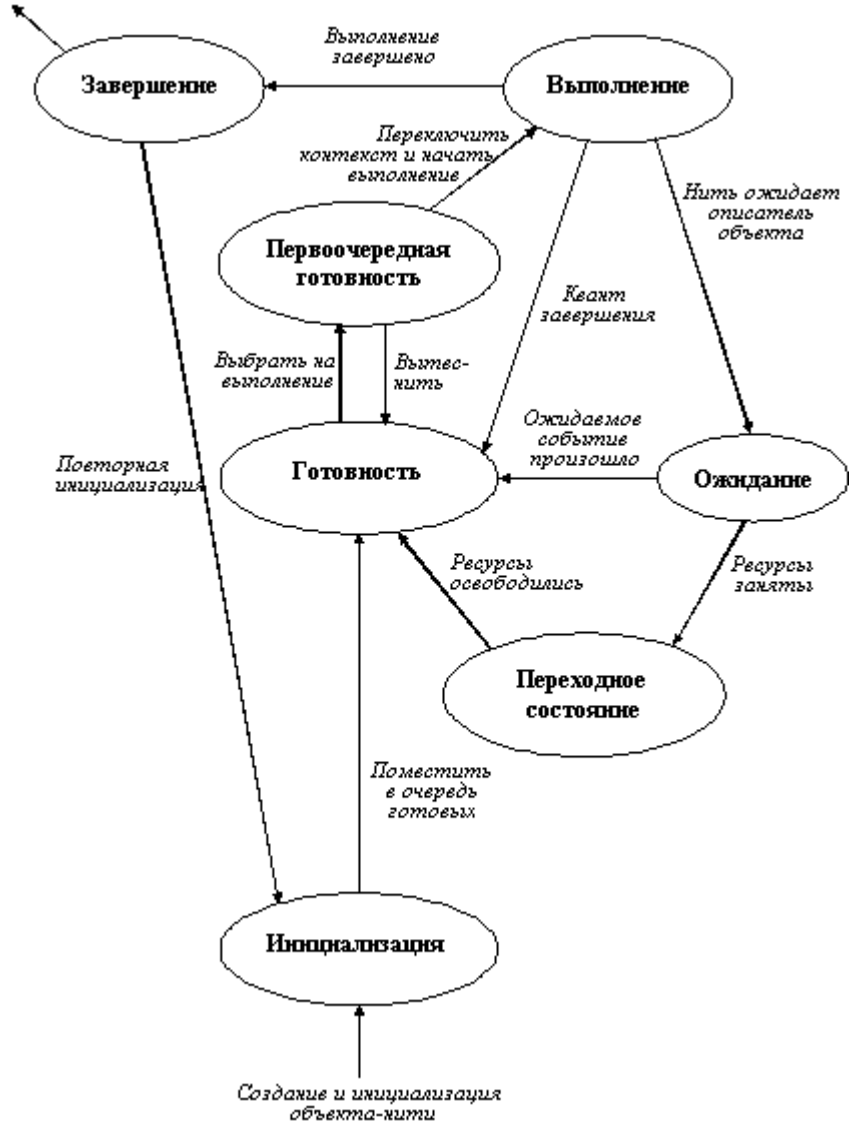


Рис. 1. Граф состояний нити

–*Готовность*. При поиске нити на выполнение диспетчер просматривает только нити, находящиеся в состоянии готовности, у которых есть все для выполнения, но не хватает только процессора.

–*Первоочередная готовность* (standby). Для каждого процессора системы выбирается одна нить, которая будет выполняться следующей (самая первая нить в очереди). Когда условия позволяют, происходит переключение на контекст этой нити.

–*Выполнение*. Как только происходит переключение контекстов, нить переходит в состояние выполнения и находится в нем до тех пор, пока либо ядро не вытеснит ее из-за того, что появилась более приоритетная нить или закончился квант времени, выделенный этой нити, либо нить завершится вообще, либо она по собственной инициативе перейдет в состояние ожидания.

–*Ожидание*. Нить может входить в состояние ожидания несколькими способами: нить по своей инициативе ожидает некоторый объект для того, чтобы синхронизировать свое выполнение; операционная система (например, подсистема ввода-вывода) может ожидать в интересах нити; подсистема окружения может непосредственно заставить нить приостановить себя. Когда ожидание нити подойдет к концу, она возвращается в состояние готовности.

–*Переходное состояние*. Нить входит в переходное состояние, если она готова к выполнению, но ресурсы, которые ей нужны, заняты. Например, страница, содержащая стек нити, может быть выгружена из оперативной памяти на диск. При освобождении ресурсов нить переходит в состояние готовности.

–*Завершение*. Когда выполнение нити закончилось, она входит в состояние завершения. Находясь в этом состоянии, нить может быть либо удалена, либо не удалена. Это зависит от алгоритма работы менеджера объектов, в соответствии с которым он и решает, когда удалять объект. Если executive имеет указатель на объект-нить, то она может быть инициализирована и использована снова.

В многопроцессорных системах при диспетчеризации и планировании нитей играет роль их процессорная совместимость: после того, как ядро выбрало нить с наивысшим приоритетом, оно проверяет, какой процессор может выполнить данную нить и, если атрибут нити "процессорная совместимость" не позволяет нити выполняться ни на одном из свободных процессоров, то выбирается следующая в порядке приоритетов нить.

Диспетчеризация процессов реального времени

Базовыми инструментами разработки сценария работы системы являются система приоритетов процессов (задач) и алгоритмы планирования (диспетчеризации) операционных системах реального времени.

В многозадачных ОС общего назначения используются, как правило, различные модификции алгоритма круговой диспетчеризации, основанные на понятии непрерывного кванта времени ("time slice"), предоставляемого процессу для работы. Планировщик по истечении каждого кванта времени просматривает очередь активных процессов и принимает решение, кому передать управление, основываясь на приоритетах процессов (численных значениях, им присвоенных). Приоритеты могут быть фиксированными или меняться со временем - это зависит от алгоритмов планирования в данной ОС, но рано или поздно процессорное время получат все процессы в системе

Алгоритмы круговой диспетчеризации неприменимы в чистом виде в операционных системах реального времени. Основной недостаток - непрерывный квант времени, в течение которого процессором владеет только один процесс. Планировщики же операционных систем реального времени имеют возможность сменить процесс до истечения "time slice", если в этом возникла необходимость.

Один из возможных алгоритмов планирования при этом "приоритетный с вытеснением". Мир операционных систем реального времени отличается богатством различных алгоритмов планирования: динамические, приоритетные, монотонные, адаптивные и пр., цель же всегда преследуется одна - предоставить инструмент, позволяющий в нужный момент времени исполнять именно тот процесс, который необходим.

Тема 7. Организация и управление памятью.

Во-первых, это осуществление контроля использования ресурса, т.е. одной из функций оперативной памяти является учет состояния каждой доступной в системе единицы: знание о том, свободна она или распределена.

Второй задачей является выбор стратегии распределения памяти. Иными словами, решается задача, какому процессу, в течение которого времени и в каком объеме должен быть выделен соответствующий ресурс. Стратегия распределения памяти является достаточно сложной задачей планирования.

Конкретное выделение ресурса тому или иному потребителю является третьей задачей управления ОЗУ. Эта подзадача следует за предыдущей задачей планирования: после решения задачи, какому процессу сколько выделить памяти и на какое время в соответствии с наличием ресурса, следует операция непосредственного выделения. Это означает, что для предоставляемого ресурса идет корректировка системных данных (например, изменение статуса занятости), а затем выдача его потребителю.

И, наконец, четвертой задачей является выбор стратегии освобождения памяти. Освобождение памяти можно рассматривать с двух точек зрения. С одной стороны, это окончательное освобождение памяти, происходящее в случае завершения процесса и высвобождения ресурса. В этом контексте задача достаточно детерминирована и не требует каких-либо алгоритмов планирования и принятия решения. С другой стороны, освобождение памяти может рассматриваться как задача принятия решения в случае, когда встает потребность высвободить физическую память из-под какого-то процесса за счет откачивания во внешнюю память, чтобы на освободившееся пространство поместить данные другого процесса. Такая задача уже не тривиальна: необходимо решить, память какого процесса необходимо откачать, какую именно область памяти у выбранного процесса будет освобождаться. В принципе можно откачать весь процесс, но это зачастую неэффективно.

Страничное распределение

Данная модель основывается на том, что все адресное пространство может быть представлено совокупностью блоков фиксированного размера, которые называются ***страницами***. Есть ***виртуальное адресное пространство*** — это то пространство, в котором оперирует программа, и ***физическое адресное пространство*** — это то пространство, которое есть в наличии у компьютера. Соответственно, при страничном распределении памяти существуют программно-аппаратные средства, позволяющие устанавливать соответствие между виртуальными и физическими страницами. Механизм преобразования виртуального адреса в физический обсуждался выше, он достаточно прост: берется номер виртуальной страницы и заменяется соответствующим номером физической страницы. Также отмечалось, что для этих целей используется т.н. ***таблица страниц***, которая целиком является аппаратной, что на самом деле является большим упрощением. Если рассмотреть современные машины с современным объемом виртуального адресного пространства, то окажется, что эта таблица будет очень большой по размеру. Соответственно, возникает важный вопрос, как осуществлять указанное отображение виртуальных адресов в физические.

Недостатком страничного распределения памяти является то, что при реализации этой модели процессу выделяется единый диапазон виртуальных адресов: от нуля до некоторого предельного значения.

Сегментное распределение

Данная модель представляет каждый процесс в виде совокупности сегментов, каждый из которых может иметь свой размер. Каждый из сегментов может иметь собственную функциональность: существуют сегменты кода, сегменты статических данных, сегмент стека и т.д. Для организации работы с сегментами может использоваться некоторая таблица, в которой хранится информация о каждом сегменте (его номер, размер и пр.). Тогда виртуальный адрес может быть проинтерпретирован, как номер сегмента и величина смещения в нем.

Модель сегментного распределения может иметь достаточно эффективно работающую аппаратную реализацию. Существует аппаратная таблица сегментов с фиксированным числом записей. Каждая запись этой таблицы соответствует своему сегменту и хранит информацию о размере сегмента и адрес начала сегмента (т.е. адрес базы), а также тут могут присутствовать различные атрибуты, которые будут оговаривать права и режимы доступа к содержимому сегмента.

К достоинствам данной модели можно отнести простоту организации, которая, по сути, явилась развитием модели распределения разделов. Если в той модели каждому процессу выделяется только один сегмент (раздел), то при сегментной модели распределения процессу выделяется совокупность сегментов, каждый из которых будет иметь свои функциональные обязанности.

К недостаткам данной модели необходимо отнести то, что каждый сегмент должен целиком размещаться в памяти (возникает упоминавшаяся выше проблема неявной неэффективности, связанная с принципом локальности). Также возникают проблемы с откачкой/подкачкой: подкачка осуществляется всем процессом или, по крайней мере, целым сегментом, что зачастую оказывается неэффективно. И поскольку каждый сегмент так или иначе должен быть размещен в памяти, то возникает ограничение на предельный размер сегмента.

Сегментно-страничное распределение

Естественным развитием рассмотренной модели сегментного распределения памяти стала модель сегментно-страничного распределения. Эта модель рассматривает виртуальный адрес, как номер сегмента и смещение в нем. Имеется также аппаратная таблица сегментов, посредством которой из виртуального адреса получается т.н. ***линейный адрес***, который, в свою очередь, представляется в виде номера страницы и величины смещения в ней. А затем, используя таблицу страниц, получается непосредственно физический адрес.

Итак, данный механизм подразумевает, что в процессе имеется ряд виртуальных сегментов, которые дробятся на страницы. Поэтому данная модель сочетает в себе, с одной стороны, логическое сегментирование, а с другой стороны, преимущества страничной организации (когда можно работать с отдельными страницами памяти, не требуя при этом полного размещения сегмента в ОЗУ).

Примером реализации может служить реализация, предложенная компанией Intel. Рассмотрим упрощенную модель этой реализации. Виртуальный адрес в этой модели представляется в виде ***селектора*** (информации о сегменте) и смещения в сегменте.

Среди особенностей данной модели можно отметить, что можно «выключать» страничную функцию, и тогда модель Intel начинает работать по сегментному распределению. А можно не использовать сегментную организацию процесса, и тогда данная реализация будет работать по страничному распределению памяти.

Тема 8. Файловая система. Управление вводом/выводом. Варианты структур ядра ОС.

Файловая система - это часть операционной системы, назначение которой состоит в том, чтобы обеспечить пользователю удобный интерфейс при работе с данными, хранящимися на диске, и обеспечить совместное использование файлов несколькими пользователями и процессами.

В широком смысле понятие "файловая система" включает:

- совокупность всех файлов на диске;
- наборы структур данных, используемых для управления файлами, такие, например, как каталоги файлов, таблицы распределения свободного и занятого пространства на диске;
- комплекс системных программных средств, реализующих управление файлами, в частности: создание, уничтожение, чтение, запись, именование, поиск и другие операции над файлами.

Файловая система FAT

Файловая система FAT (таблица размещения файлов) представляет собой простую файловую систему, разработанную для небольших дисков и простых структур каталогов. В целях защиты тома (логический диск) на нем хранятся две копии FAT, на тот случай, если одна из них окажется поврежденной.
Том, отформатированный для использования файловой системы FAT, размечается по кластерам. Кластер – минимальная адресуемая единица дисковой памяти, выделяемая файлу. Размер кластера по умолчанию определяется размером тома. При использовании файловой системы FAT номер кластера должен иметь длину не более 16 бит и представлять собой одну из степеней 2.

Структура тома FAT.

Загрузочный сектор является первым на логическом диске и состоит из:

1. DPB – блок параметров диска. Служит для идентификации физических параметров логического диска.
2. SB – загрузчик.

Таблицы расположения файлов (области FAT1 и FAT2) содержат следующую информацию о каждом кластере тома:

- Unused (кластер не используется)
- Cluster in use by a file (кластер используется файлом)
- Bad cluster (плохой кластер)
- Last cluster in a file (последний кластер файла)

Корневой каталог содержит записи для каждого файла и каждого каталога, расположенных в корневом каталоге. Единственным различием между корневым каталогом и всеми остальными каталогами является то, что корневой каталог занимает четко определенное место на диске и имеет фиксированный размер (512 записей для жесткого диска; для дискет этот размер определяется объемом дискеты).

Каталоги содержат 32-байтные записи для каждого содержащегося в них файла и каждого вложенного каталога. Эти записи содержат следующую информацию:

- имя (в формате "8+3"),
- байт атрибутов (8 бит),
- время создания (24 бит),
- дата создания (16 бит),
- дата последнего доступа (16 бит),
- время последней модификации (16 бит),
- дата последней модификации (16 бит),
- номер начального кластера файла в таблице расположения файлов (16 бит),
- размер файла (32 бит).

Структура каталога FAT не имеет четкой организации, и файлам присваиваются первые доступные адреса кластеров на томе. Номер начального кластера файла представляет собой адрес первого кластера, занятого файлом, в таблице расположения файлов. Каждый кластер содержит указатель на следующий кластер, использованный файлом, или индикатор (OxFFFF), указывающий на то, что данный кластер является последним кластером файла.

Файл FAT имеет 4 атрибута, которые могут сбрасываться и устанавливаться пользователем:

- archive file (архивный файл),
- system file (системный файл),

- hidden file (скрытый файл),
- read-only file (файл только для чтения).

ОС поддерживающие FAT: MS DOS, Windows 95, Windows NT, OS/2.

Файловая система NTFS

NTFS разработана для:

- быстрого выполнения стандартных файловых операций чтения, записи и поиска;
- быстрого выполнения улучшенных операций типа восстановления файловой системы на очень больших жестких дисках.

Структура тома NTFS

MFT	Зона MFT	Зона для размещения файлов и каталогов	Копия первых 16 записей MFT	Зона для размещения файлов и каталогов

Диск NTFS делится на две части:

1. первые 12% диска отводятся под MFT зону - пространство, в которое растет метафайл MFT. Запись каких-либо данных в эту область невозможна. MFT-зона всегда держится пустой - это делается для того, чтобы самый главный, служебный файл (MFT) не фрагментировался при своем росте;
2. остальные 88% диска представляют собой обычное пространство для хранения файлов.

MFT – метафайл – специальный файл, позволяющий определить местонахождение всех остальных файлов.

Метафайлы имеют строго фиксированное положение. Их копия содержится в середине для надёжности.

MFT находятся в корневом каталоге NTFS диска.

NTFS просматривает каждый файл (или каталог) как набор атрибутов файла. Такие элементы, как имя файла, информация зашиты и даже данные — все это атрибуты файла. Каждый атрибут идентифицирован кодом типа атрибута и, необязательно, именем атрибута.

Имя файла может содержать любые символы. Максимальная длина 256 символов. Каталог в NTFS представляет собой специальный файл, хранящий ссылки на другие файлы и каталоги, создается иерархическое строение данных. Он поделён на блоки, каждый из которых содержит имя файла, базовые атрибуты и ссылку на элемент MFT.

Управление вводом/выводом

Программирование ввода-вывода является наиболее сложным и трудоемким, требующим очень высокой квалификации. Поэтому код, реализующий операции ввода-вывода, сначала стали оформлять в виде системных библиотечных процедур, а потом и вовсе вывели из систем программирования, включив в операционную систему. Это позволило не писать такой код в каждой программе, а только обращаться к нему — системы программирования стали генерировать обращения к системному коду ввода-вывода. Таким образом, ***управление вводом- выводом — это одна из основных функций*** любой операционной системы.

Основные концепции организации ввода-вывода в операционных системах.

Система ввода-вывода, способная объединить в одной модели широкий спектр устройств, должна быть универсальной. Она должна учитывать потребности существующих устройств, от простой мыши до клавиатур, принтеров, графических дисплеев, дисковых накопителей, компакт-дисков и даже сетей. С другой стороны, необходимо обеспечить доступ к устройствам ввода-вывода для множества параллельно выполняющихся задач, причем так, чтобы они как можно меньше мешали друг другу.

Поэтому самым главным является следующий **принцип**: *любые операции по управлению вводом-выводом объявляются привилегированными и могут выполняться только кодом самой операционной системы.* Для обеспечения этого принципа в большинстве процессоров даже вводятся *режимы пользователя* и *супервизора.* Последний еще называют *привилегированным режимом,* или *режимом ядра.* Как правило, в режиме *супервизора* выполнение команд ввода-вывода разрешено, а в пользовательском режиме — запрещено. Обращение к командам ввода-вывода в пользовательском режиме вызывает *исключение1,* и управление через механизм прерываний передается коду операционной системы. Хотя возможны и более сложные схемы, в которых в ряде случаев пользовательским программам может быть разрешено непосредственное выполнение команд ввода-вывода.

Помимо разделяемых устройств ввода-вывода (эти устройства допускают разделение посредством механизма доступа) существуют неразделяемые устройства. ***Примерами разделяемого устройства*** могут служить накопитель на магнитных дисках, устройство чтения компакт-дисков. Это устройства с прямым доступом. ***Примеры неразделяемых устройств*** — принтер, накопитель на магнитных лентах. Это устройства с последовательным доступом. Операционные системы должны

управлять и теми и другими, предоставляя возможность параллельно выполняющимся задачам их использовать.

Итак, управление вводом-выводом осуществляется компонентом операционной системы, который часто называют ***супервизором ввода-вывода.*** Перечислим основные ***задачи,***возлагаемые на супервизор.

1. Модуль супервизора операционной системы, иногда называемый *супервизором задач,* получает запросы от прикладных задач на выполнение тех или иных операций, в том числе на ввод-вывод. Эти запросы проверяются на корректность и если они соответствуют спецификациям и не содержат ошибок, то обрабатываются дальше. В противном случае пользователю (задаче) выдается соответствующее диагностическое сообщение о недействительности (некорректности) запроса.
2. Супервизор ввода-вывода получает запросы на ввод-вывод от супервизора задач или от программных модулей самой операционной системы.
3. Супервизор ввода-вывода вызывает соответствующие распределители каналов и контроллеров, планирует ввод-вывод (определяет очередность предоставления устройств ввода-вывода задачам, затребовавшим эти устройства).
4. Супервизор ввода-вывода инициирует операции ввода-вывода (передает управление соответствующим драйверам) и в случае управления вводом-выводом с использованием прерываний предоставляет процессор диспетчеру задач с тем, чтобы передать его первой задаче, стоящей в очереди на выполнение.
5. При получении сигналов прерываний от устройств ввода-вывода супервизор идентифицирует эти сигналы и передает управление соответствующим программам обработки прерываний.
6. Супервизор ввода-вывода осуществляет передачу сообщений об ошибках, если таковые происходят в процессе управления операциями ввода-вывода.
7. Супервизор ввода-вывода посылает сообщения о завершении операции ввода-вывода запросившей эту операцию задаче и снимает ее с состояния ожидания ввода-вывода, если задача ожидала завершения операции.

Режимы управления вводом-выводом

Как известно, имеется два основных режима ввода-вывода: *режим обмена с опросом готовности* устройства ввода-вывода и *режим обмена с прерываниями.*

Пусть для простоты рассмотрения этих вопросов управление вводом-выводом осуществляет центральный процессор. В этом случае часто говорят о работе *программного канала* обмена данными между внешними устройством и оперативной памятью (в отличие от *канала прямого доступа к памяти,* при котором управление вводом-выводом осуществляет специальное дополнительное оборудование).

Закрепление устройств, общие устройства ввода-вывода.

Как известно, многие устройства и, прежде всего, устройства с последовательным доступом не допускают совместного использования. Такие устройства могут стать*закрепленными* за процессом, то есть их можно предоставить некоторому вычислительному процессу на все время жизни этого процесса. Однако это приводит к тому, что вычислительные процессы часто не могут выполняться параллельно они ожидают освобождения устройств ввода-вывода. Чтобы организовать совместное использование многими параллельно выполняющимися задачами тех устройств ввода-вывода, которые не могут быть разделяемыми, вводится понятие *виртуальных устройств.* Принцип виртуализации позволяет повысить эффективность вычислительной системы.

Вообще говоря, понятие виртуального устройства шире, нежели понятие *спулинга*

Основное назначение спулинга — создать видимость разделения устройства ввода-вывода, которое фактически является устройством с последовательным доступом и должно использоваться только монопольно и быть закрепленным за процессом. Например, мы уже говорили, что в случае, когда несколько приложений должны выводить на печать результаты своей работы, если разрешить каждому такому приложению печатать строку по первому же требованию, то это приведет к потоку строк непредставляющих никакой ценности. Однако если каждому вычислительному процессу предоставлять не реальный, а виртуальный принтер, и поток выводимых символов (или управляющих кодов для их печати) сначала направлять в специальный файл на диске (так называемый *спул-файл* — spool-file) и только потом, по

окончании виртуальной печати, в соответствии с принятой дисциплиной обслуживания и приоритетами приложений выводить содержимое спул-файла на принтер, то все результаты работы можно будет легко читать. Системные процессы, которые управляют спул-файлом, называются *спулером чтения* (spool-reader) или *спулером записи* (spool-writer).

Различает термины ≪принтер≫ и ≪устройство печати≫. Принтер — это некоторая виртуализация, объект операционной системы, а устройство печати — это физическое устройство, которое может быть подключено к компьютеру. Принтер может быть *локальным* или *сетевым.*

При установке локального принтера в операционной системе создается новый объект, связанный с реальным устройством печати через тот или иной интерфейс.

Если же некоторый локальный принтер предоставить в сети в общий доступ с теми или иными разрешениями, то для других компьютеров и их пользователей он может стать*сетевым,* Компьютер, на котором имеется локальный принтер, предоставленный в общий доступ, называется *принт-сервером.*

Варианты структур ядра ОС

По основному архитектурному принципу ОС разделяются на *микроядерные* и *монолитные*. В качестве примера микроядерной ОС привести ОСРВ QNX, в качестве монолитной можно назвать ОС семейства Windows, ОС Linux. Ядро ОС Windows пользователь не может изменить, т.к. не доступны его исходные коды и нет программы для сборки (компиляции) этого ядра. В случае с Linux пользователь может сам собрать ядро, которое необходимо, включив в него те необходимые программные модули и драйверы, которые считает целесообразным включить именно в ядро (а не обращаться к ним из ядра).

Микроядерные операционные системы

Микроядро – это минимальная стержневая часть операционной системы, служащая основой модульных и переносимых расширений.

Основная идея, заложенная в технологию микроядра, заключается в том, чтобы конструировать необходимую среду верхнего уровня, из которой можно легко получить доступ ко всем функциональным возможностям уровня аппаратного обеспечения. При такой структуре ядро служит стартовой точкой для создания системы. В микроядре содержится и исполняется минимальное количество кода, необходимое для реализации основных системных вызовов. В число этих вызовов входят передача сообщений и организация другого общения между внешними по отношению к микроядру процессами, поддержка управления прерываниями, а также ряд некоторых других функций. Остальные функции, характерные для «обычных» (не микроядерных) ОС, обеспечиваются как модульные дополнения-процессы, взаимодействующие главным образом между собой и осуществляющие взаимодействие посредством передачи сообщений.

Микроядро является маленьким, передающим сообщения модулем системного программного обеспечения, работающим в наиболее приоритетном состоянии компьютера и поддерживающим остальную часть операционной системы, рассматриваемую как набор серверных приложений.

Созданная в университете Карнеги Меллон технология микроядра Mach служит основой для многих ОС.

Исполняемые микроядром функции ограничены в целях сокращения его размеров и максимизации количества кода, работающего как прикладная программа. Микроядро включает только те функции, которые требуются для определения набора абстрактных сред обработки для прикладных программ и для организации совместной работы приложений в обеспечении сервисов и в действии клиентами и серверами. В результате микроядро обеспечивает только пять различных типов сервисов:

управление виртуальной памятью;
задания и потоки;
межпроцессные коммуникации (IPC[1]);
управление поддержкой ввода/вывода и прерываниями;
сервисы набора процессора.

Другие подсистемы и функции операционной системы, такие как системы управления файлами, поддержка внешних устройств и традиционные программные интерфейсы, размещаются в одном или более системных сервисах либо в задаче операционной системы. Эти программы работают как приложения микроядра.

Благодаря своим размерам и способности поддерживать стандартные сервисы программирования и характеристики в виде прикладных программ сами микроядра проще, чем ядра монолитных операционных систем. С микроядром функция операционной системы разбивается на модульные части, которые могут быть сконфигурированы целым рядом способов, позволяя строить большие системы добавлением частей к меньшим. В некоторых случаях определенной сложностью использования микроядерного подхода на практике является замедление скорости выполнения системных вызовов при передаче сообщений через микроядро по сравнению с классическим подходом. С другой стороны, можно констатировать и обратное. Поскольку микроядра малы и имеют сравнительно мало требуемого к исполнению кода уровня ядра, они обеспечивают удобный способ поддержки характеристик реального времени, требующихся для мультимедиа, управления устройствами и высокоскоростных коммуникаций. Хорошо структурированные микроядра обеспечивают изолирующий слой для аппаратных различий, которые не маскируются применением языков программирования высокого

уровня. Таким образом, они упрощают перенесение кода и увеличивают уровень его повторного использования.

Наиболее ярким представителем микроядерных ОС является ОС реального времени QNX. Микроядро QNX поддерживает только планирование и диспетчеризацию процессов, взаимодействие процессов, обработку прерываний и сетевые службы нижнего уровня. Микроядро обеспечивает всего лишь пару десятков системных вызовов, но благодаря этому оно может быть целиком размещено во внутреннем кэше даже таких процессоров, как Intel 486. Разные версии этой ОС имели и различные объемы ядер – от 8 до 46 Кбайт.

Микроядерная архитектура является альтернативой классическому способу построения ОС. Под классической архитектурой в данном случае понимается структурная организация ОС, в соответствии с которой все основные функции ОС выполняются в привилегированном режиме.

Для надежного управления ходом выполнения приложений операционная система должна иметь по отношению к приложениям определенные привилегии. Иначе некорректно работающее приложение может вмешаться в работу ОС и, например, разрушить часть ее кодов.

Обеспечить привилегии операционной системе невозможно без специальных средств аппаратной поддержки. Аппаратура компьютера должна поддерживать как минимум два режима работы — пользовательский режим (user mode) и привилегированный режим, который также называют режимом ядра (kernel mode), или режимом супервизора (supervisor mode). Подразумевается, что операционная система или некоторые ее части работают в привилегированном режиме, а приложения – в пользовательском режиме.

При этом некоторые вспомогательные функции ОС оформляются в виде приложений и выполняются в пользовательском режиме наряду с обычными пользовательскими программами (становясь системными утилитами или обрабатывающими программами). Каждое приложение пользовательского режима работает в собственном адресном пространстве и защищено тем самым от какого-либо вмешательства других приложений. Код ядра, выполняемый в привилегированном режиме, имеет доступ к областям памяти всех приложений, но сам полностью от них защищен. Приложения обращаются к ядру с запросами на выполнение системных функций.

Суть микроядерной архитектуры состоит в следующем. В привилегированном режиме остается работать только очень небольшая часть ОС, называемая микроядром (см. рис. 13.1). Микроядро защищено от остальных частей ОС и приложений. В состав микроядра обычно входят машинно-зависимые модули, а также модули, выполняющие базовые функции ядра по управлению процессами, обработке прерываний, управлению виртуальной памятью, пересылке сообщений и управлению устройствами ввода-вывода, связанные с загрузкой или чтением регистров устройств. Набор функций микроядра обычно соответствует функциям слоя базовых механизмов обычного ядра. Такие функции ОС трудно выполнить в пространстве пользователя.

Мультипроцессорные ОС. Сетевые ОС. Распределенные ОС.
Мультипроцессорные ОС
Сетевые ОС
Распределенные ОС

Тема 9. Мультипроцессорные ОС. Сетевые ОС. Распределенные ОС.

Мультипроцессорные ОС

Важным, но часто не слишком очевидным расширением однопроцессорных операционных систем является возможность поддержки нескольких процессоров, имеющих доступ к совместно используемой памяти. Концептуально это расширение несложно. Все структуры данных, необходимые операционной системе для поддержки аппаратуры, включая поддержку нескольких процессоров, размещаются в памяти. Основная разница заключается в том, что теперь эти данные доступны нескольким процессорам и должны быть защищены от параллельного доступа для обеспечения их целостности.

Однако многие операционные системы, особенно предназначенные для персональных компьютеров и рабочих станций, не могут с легкостью поддерживать несколько процессоров. Основная причина такого поведения состоит в том, что они были разработаны как монолитные программы, которые могут выполняться только в одном потоке управления. Адаптация таких операционных систем под мультипроцессорные означает повторное проектирование и новую реализацию всего ядра. Современные операционные системы изначально разрабатываются с учетом возможности работы в мультипроцессорных системах.

Многопроцессорные операционные системы нацелены на поддержание высокой производи-

тельности конфигураций с несколькими процессорами. Основная их задача — обеспечить прозрачность числа процессоров для приложения. Сделать это достаточно легко, поскольку сообщение между различными приложениями или их частями требует тех же примитивов, что и в многозадачных однопроцессорных операционных системах. Идея состоит в том, что все сообщение происходит путем работы с данными в специальной совместно используемой области данных, и все что нам нужно — это защитить данные от одновременного доступа к ним. Защита осуществляется посредством примитивов синхронизации.

Сетевые ОС.

В зависимости от того, какой виртуальный образ создает ОС для того, чтобы подменить им реальную аппаратуру персональной сети разделяют:

а) сетевой ОС (СОС),

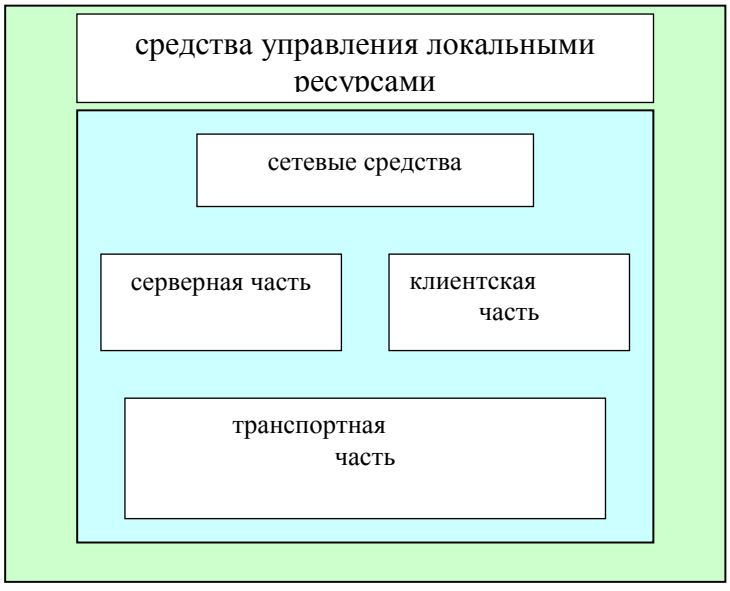
б) распределенная ОС (РОС)

СОС представляет пользователю некую виртуальную вычислительную систему, работа с которой гораздо проще, чем с реальной сетевой аппаратурой. В то же время эта виртуальная система не полностью скрывает распределенную природу своего реального прототипа, т.е. является виртуальной сетью. При использовании ресурсов персональной сети пользователь СОС всегда понимает, что он имеет дело с сетевыми ресурсами. Он должен быть в курсе того, где хранятся его файлы и должен использовать явные команды передачи файлов для перемещения их с одной машины на другую. В среде СОС пользователь может запустить задание на любой ЭВМ сети, но всегда должен знать на какой машине выполняется его задание. По умолчанию пользовательское задание выполняется на той ЭВМ, на которой был сделан логический вход. Магистральным направлением развития СОС является достижение как можно большей степени прозрачности сетевых ресурсов. В идеальном случае СОС должен предоставлять системные ресурсы в виде ресурсов единой централизованной виртуальной машины. Для такой ОС используют специальное назначение. РОС или истинно РОС. РОС динамически и автоматически распределяет работу по различным машинам сигналом для обработки, заставляет набор сетевых машин работать как виртуальный уни-процессор. Пользователь РОС не имеет сведений о том, на какой машине выполняется работа. РОС существует как единая ОС в масштабах вычислительной системы. В настоящее время практически все СОС далеки от идеала истинной распределенности, поэтому под термином СОС понимают 2 значения:

a) Как совокупность всех компонентов, т.е. на разных ЭВМ могут быть одинаковые или разные ОС. Все ОС будут функционировать независимо друг от друга каждая из них будет принимать решение о создании и завершении своих собственных процессов и управлений локальными ресурсами, но при этом все ЭВМ должны включать взаимосогласованный набор коммуникационных протоколов для организации взаимодействующих процессов.

b) Как ОС отдельного компьютера, способного работать в сети, т.е. если ОС отдельного компьютера позволяет ему работать в сети, т.е. предоставляет свои ресурсы в общее пользование и (или) потреблять ресурсы других ЭВМ в сети.

Функциональные компоненты СОС



СОС состоит:

а) средства управления локальными ресурсами – реализуют все функции ОС автономного компьютера (распределение ОП, планирование и диспетчеризация процессов, пользовательский интерфейс).
б) сетевые ресурсы, которые делятся на

- серверная часть ОС – это средства предоставления локальных услуг и ресурсов в общее пользование.
- клиентская часть ОС – это средства запросов доступа к удаленным ресурсам и услугам.
- транспортные средства – это средства, которые совместно с коммуникацией сетей обеспечивают передачу сообщений между ПК сети.

Распределенные ОС.

Существует два типа распределенных операционных систем. *Мультипроцессорная операционная система (multiprocessor operating system)* управляет ресурсами мультипроцессора. *Мультикомпъютерная операционная система (multicomputeroperating system)* разрабатывается для гомогенных мультикомпьютеров. Функциональность распределенных операционных систем в основном не отличается от функциональности традиционных операционных систем, предназначенных для компьютеров с одним процессором за исключением того, что она поддерживает функционирование нескольких процессоров. Поэтому давайте кратко обсудим операционные системы, предназначенные для обыкновенных компьютеров с одним процессором.

Тема 10. Вычислительный процесс. Обслуживание прерываний

Система прерывания

Во время выполнения компьютером текущей программы, внутри компьютера, а также во внешней среде, связанной с ним (клавиатура, дисплей, внешняя память, технологические и научно-исследовательские процессы, управляемые компьютером и т.д.) могут возникать события, требующие немедленной реакции на них с его стороны. Реакция компьютера заключается в том, что он прерывает обработку текущей программы и переходит к выполнению некоторой другой программы (подпрограммы, процедуры), специально предназначенной для выполнения требований, запрашиваемых данным событием. Процесс выполнения такой подпрограммы называется **обработкой (обслуживанием)** данного прерывания, а сама подпрограмма обслуживания прерывания часто называется **обработчиком** прерывания.

Итак:

Прерыванием называется процесс, обуславливающий реакцию компьютера на некоторое событие, требующее немедленного его вмешательства.

Запрос прерывания – есть сигнал процессору о появлении события, требующего немедленной реакции процессора путем перехода на подпрограмму обслуживания этого события.

Системой прерывания называется комплекс аппаратных и программных

средств, обеспечивающих выявление запросов прерывания и эффективное их обслуживание.

Основными функциями системы прерывания являются следующие.

1. Запоминание состояния процессора, которое определяется состоянием его основных регистров на момент прерывания текущей программы.

2. Осуществление передачи управления на процедуру обслуживания прерывания.

3. Восстановление состояния процессора после окончания выполнения процедуры обслуживания прерывания.

4. Передача управления на продолжение выполнения текущей программы.

Таким образом, после появления запроса прерывания и возможности осуществления самого процесса прерывания, процессор прерывает выполнение текущей программы, автоматически запоминая в стеке адрес возврата (CS и IP), а также содержимое регистра флагов Flags. В стек может, также, передаваться и сохраняться содержимое тех регистров, которые могут понадобиться при исполнении процедуры обслуживания прерывания и, следовательно, могут быть искажены.

Классификация типов прерывания.

Запросы на прерывание могут возникать как внутри самого процессора, так и со стороны внешней среды (периферийных устройств). Поэтому можно выделить сразу же два класса типов прерываний.

1. Внешние прерывания.

2. Внутренние прерывания.

Здесь: **INTi (Interrupt i) –** i-ый источник (тип) запроса прерывания от внешних устройств;

NMI (Nonmaskable Interrupt) – немаскируемое прерывание, т.е. прерывание, которое нельзя замаскировать;

INTR (Interrupt) – общий запрос на прерывание процессора;

INTA (Interrupt Acknowledge) – подтверждение процессором получения запроса прерывания, и готовность принятия кода типа прерывания.

Внешние прерывания

Внешние прерывания относят к так называемым **аппаратным** прерываниям, поскольку они вызываются аппаратурой, находящейся вне пределов самого компьютера. Внешние прерывания подразделяются на:

а) маскируемые прерывания,

б) немаскируемые прерывания.

Маскируемые прерывания.

Маскируемыми – называются такие прерывания, запросы которых могут быть выключены из зоны внимания процессора, путем наложения **маски** на возможные запросы прерываний.

Запросы маскируемых прерываний, в отличие от остальных, поступают на систему прерываний процессора через специальное устройство, называемое **контроллером прерываний**, который в современных процессорах входит в состав микросхемы «южного моста» Chipset, размещаемой на системной плате компьютера. Основные функции контроллера прерываний заключаются в том, что с его помощью можно:

- вследствие различной важности устройств, которые могут запрашивать прерывания у компьютера, устанавливать запросам прерывания определенный *уровень приоритета*,при котором очередность их обработки осуществляется строго в порядке присвоенной им приоритетности. Это дает возможность запросу с большим уровнем приоритета прерывать исполнение процедур обслуживания прерывания от устройств с меньшим уровнем приоритета;

- устанавливать определенный порядок обслуживания запросов прерываний одного уровня привилегий, при одновременном их появлении;

- дать возможность пользователю, при необходимости, запрещать компьютеру, реагировать на запросы прерывания внешних устройств, как всех одновременно, так и по отдельности для каждого устройства (т.е. выполнять так называемое маскирование запросов прерывания). Запрещение или разрешение индивидуальных запросов прерываний реализуется при помощи использования в контроллерах прерываний кода маски.

Маска представляет собой двоичный код, каждый разряд которого соответствует некоторому запросу прерывания. Обычно устройство маскирования делают таким образом, чтобы логическая единица, записанная в разряд маски, запрещала прохождение соответствующего запроса прерывания к процессору, а логический ноль – разрешала.

Заметим при этом, что маскируемые прерывания, все вместе одновременно, могут быть запрещены или разрешены, путем установки или сброса флага IF в регистре флагов процессора, подачей соответствующих команд STI или CLI.

Немаскируемые прерывания.

Запросы немаскируемых прерываний подаются на специально выделенный вход NMI. Они запрещены быть не могут. Источниками запросов таких прерываний являются, например, устройства контроля пропадания напряжения питания или контроля повреждения каких-либо особо важных систем обработки.

Все внешние прерывания асинхронны по отношению к работе процессора, так как они могут появиться в любой момент времени. И, поскольку начало передачи управления на подпрограмму обработки прерывания происходит только после окончания исполнения команды, то *время реакции*, т.е. запаздывания начала обслуживания по отношению к появлению запроса прерывания, может быть, при выполнении, например, команд умножения или деления, достаточно велико.

Внутренние прерывания.

Внутренние прерывания образуются внутри компьютера по запросам его отдельных функциональных узлов, или как следствие выполнения некоторых специальных команд INT N. Внутренние прерывания замаскированы быть не могут. Код типа прерываний такого рода формируется автоматически при поступлении соответствующего запроса внутреннего прерывания.

Внутренние прерывания бывают двух типов:

а) **особые случаи или исключения;**

б) **программные прерывания.**

Особые случаи или исключения.

Особые случаи или исключения – это прерывания, которые возникают при аномальной ситуации, сложившейся при выполнении конкретной команды и препятствующей нормальному продолжению программы.

К прерываниям – особым случаям (исключениям), относятся, например, такие ситуации:

- ошибка деления (частный случай – деление на 0);

- переполнение разрядной сетки после выполнения очередной команды;

- неразрешенный код команды;

- пошаговое прерывание, которое включается при установке в состояние 1 флага TF в регистре флагов Flags. При этом после выполнения каждой команды программы, в стеке автоматически запоминается состояние CS, IP, Flags и выводится на экран дисплея состояние всех регистров процессора и части ячеек ОЗУ. После этого процесс выполнения текущей программы останавливается и ожидается следующий запуск программы для выполнения очередного шага; и т.п.

Программные прерывания.

Под **программными прерываниями** понимаются прерывания, инициируемые самой выполняемой программой, а именно, командами типа INT N, где N – номер прерывания в некотором заданном диапазоне.

Действие этих прерываний весьма похожи на вызов подпрограмм (процедур), за исключением того, что передача управления осуществляется не по адресу, указываемому в команде CALL, а по номеру (типу) прерывания, по которому в таблице векторов прерывания находится адрес передачи управления. Кроме того, при программном прерывании обязательно автоматически запоминается в стеке состояние регистра флагов Flags. Программные прерывания отличаются от других тем, что они задаются заранее и определены в программе, а не являются случайными по отношению к работе компьютера. Они широко используются при вызове системных функций DOS и BIOS.

Основные характеристики систем прерывания.

К основным характеристикам систем прерывания относятся.

а) **Общее количество типов запросов прерывания** (входов в систему прерывания).

б) Время реакции – интервал времени между появлением запроса прерывания и началом выполнения действий по обслуживанию прерывания.

в) Издержки прерывания – затраты времени на переключение программ .

г) Глубина прерывания – максимальное число программ, которые могут прерывать друг друга Если после перехода к процедуре обслуживания прерывания и вплоть до ее окончания прием других запросов запрещается, то говорят, что система прерываний имеет глубину прерывания равную 1. Глубина прерывания равна **n**, если допускается последовательное прерывание до **n** программ. Последовательность прерывания программ определяется заданными уровнями их приоритетов. Глубина системы прерывания обычно совпадает с числом уровней приоритета в системе прерываний.

д) Насыщение системы прерываний – насыщением системы прерывания называется ситуация, когда запрос прерывания окажется необслуженным к моменту прихода нового запроса от того же источника (этот случай обозначен пунктиром). В этом случае, предыдущий запрос прерывания от того же источника будет утрачен, что недопустимо. Поэтому быстродействие компьютера, характеристики системы прерывания, число источников прерывания и частота возникновения запросов должны быть согласованы таким образом, чтобы насыщение было невозможным.

Классификация систем прерывания.

Идентификация запросов прерывания (определение источников запросов) в системах прерывания может осуществляться двумя основными способами: способом опроса (Polling) и векторным способом.

В связи с этим и системы прерывания могут быть реализованы двух типов:

1. **Системы прерывания с опросом источников прерывания (Polling).**

2. **Векторные системы прерывания.**

Следует, однако, заметить, что в современных компьютерах системы прерывания строятся как комбинированные системы, поскольку они могут работать как в режиме опроса, так и в режиме векторного запроса, причем режим опроса используется сейчас довольно редко.

Для упрощения понимания рассмотрим эти режимы по отдельности.

Системы прерывания с опросом – поллинг (polling).

Системы прерывания с опросом характерны тем, что опрос процессором состояний входов запросов прерывания осуществляется в удобное для него время (например, в промежутках между выполнением программ). Отсюда следует, что главным отличием систем прерывания с опросом является то, что инициатива идентификации и обслуживания запросов прерываний от периферийных устройств, принадлежит *процессору*.

Различают две разновидности режима опроса:

а) программный поллинг;

б) аппаратный поллинг.

а) Программный поллинг.

При программном поллинге запросы прерывания опрашиваются программно, так же программно определяется порядок их опроса и их обслуживания. Схема алгоритма реализации программного поллинга приведена на рис XI.6. Появление самих запросов прерывания фиксируется в

специальном регистре, так называемом регистре*обслуживаемых запросов прерываний*, разряды которого часто именуются «флагами», аналогично разрядам регистра Flags процессора. Регистр обслуживаемых запросов прерываний находится в контроллере прерываний, который, в свою очередь, как уже упоминалось выше, в современных процессорах входит в состав «южного» моста Chipset. Приход запроса прерывания устанавливает соответствующий ему бит этого регистра в состояние 1 (устанавливает «флаг» появления запроса).

б) Аппаратный поллинг.

При аппаратном поллинге, определение порядка обслуживания запросов прерывания осуществляется аппаратным образом. Например, используя схему

При этом линия сигнала запроса прерывания INTR# является общей для всех устройств, а линия сигнала подтверждения прерывания INTA соединяет устройства таким образом, что сигнал проходит через каждое из них по очереди. Таким образом, когда появляется сигнал запроса прерывания INTR# от одного или нескольких устройств, то генерируемый процессором сигнал подтверждения получения сигнала запроса прерывания и согласия на его обработку INTA, проходит последовательно через все устройства. Первым

этот сигнал получает устройство 1. Если обслуживание ему не требуется, он пересылает сигнал устройству 2.

Если же устройство 1 запрашивало прерывание и ждет ответа, оно блокирует сигнал INTA и сообщает свой идентификационный код на линии данных. Таким образом, в данной схеме соединения устройств, наивысший приоритет имеет устройство, которое ближе всего расположено к процессору с точки зрения схемы подключения.

Векторные системы прерывания.

В случае векторных систем прерывания (основных у современных процессоров) инициатива принадлежит устройству, запрашивающему прерывание. Аналогично аппаратному поллингу, при поступлении в систему хотя бы одного запроса, процессору выставляется сигнал общего прерывания – INTR. Однако, процессор, принудительно проверяет состояние этого сигнала в конце каждого машинного цикла исполнения очередной команды. Если INTR=1, и процессору разрешено прерывание (флаг IF=1), он заканчивает выполнение текущей команды и подтверждает прием запрашиваемого прерывания путем подачи на контроллер прерывания сигнал INTA (Interrupt Acknowledge). Получив сигнал INTA, контроллер выдает процессору код типа прерывания (номер прерывания), по которому процессор определяет логический адрес начала процедуры обслуживания прерывания и, который, обычно, называется **вектором прерывания .**

Поскольку полный логический адрес для процессоров i8086/88, а также для всех последующих моделей процессоров семейств X86, Pentium и Intel Core в реальном режиме работы состоит из четырех байт (два байта – указатель базового адреса сегмента и два байта – смещение в данном сегменте), то вектора прерываний являются 4-х байтовым двоичным словом.

Вектора прерываний в защищенном режиме работы всех современных компьютеров семейств X86, Pentium, Intel Core 2, 3, 5, 7 имеют размер 8 байт, поскольку они определяют специальные 8-ми байтовые структуры – дескрипторы, из которых находятся начальные физические адреса, соответствующих процедур обслуживания прерываний. И в том и в другом случае всего определено 256 типов прерываний и, следовательно, любое возможное прерывание определяется 8-ми разрядным двоичным кодом типа прерывания.

Тема 11. Многозадачные и многопользовательские ОС. Распределение ресурсов в ОС.

Многозадачные и многопользовательские ОС

Операционные системы могут различаться особенностями реализации внутренних алгоритмов управления основными ресурсами компьютеров (процессорами, памятью, устройствами), особенностями использованных методов проектирования, типами аппаратных платформ, областями использования и многими другими свойствами.

По числу одновременно работающих пользователей ОС делятся на однопользовательские и многопользовательские. Главным отличием многопользовательских систем от однопользовательских является наличие средств защиты информации каждого пользователя от несанкционированного доступа других пользователей. Следует заметить, что не всякая многозадачная система является многопользовательской, и не всякая однопользовательская ОС является однозадачной.

По числу одновременно выполняемых задач операционные системы могут быть разделены на два класса: однозадачные и многозадачные.

Однозадачные ОС выполняют функцию предоставления пользователю виртуальной машины, делая более простым и удобным процесс взаимодействия пользователя с компьютером. Однозадач-

ные ОС включают средства управления периферийными устройствами, средства управления файлами, средства общения с пользователем.

Многозадачные ОС, кроме вышеперечисленных функций, управляют разделением совместно используемых ресурсов, таких, как процессор, оперативная память, файлы и внешние устройства.

Вытесняющая и невытесняющая многозадачность. Среди множества существующих вариантов реализации многозадачности можно выделить две группы алгоритмов: невытесняющей многозадачности (NetWare, Windows 3.x) и вытесняющей многозадачности (Windows NT, OS/2, UNDC). Основным различием между вытесняющим и невытесняющим вариантами многозадачности является степень централизации механизма планирования процессов. В первом случае механизм планирования процессов целиком сосредоточен в операционной системе, а во втором – распределен между системой и прикладными программами. При невытесняющей многозадачности активный процесс выполняется до тех пор, пока он сам не отдаст управление операционной системе для того, чтобы та выбрала из очереди другой готовый к выполнению процесс. При вытесняющей многозадачности решение о переключении процессора с одного процесса на другой принимается операционной системой, а не самим активным процессом.

Распределение ресурсов в ОС.

Среди множества и разнообразия целей, которые должна выполнять любая ОС, есть одна общая - обеспечить эффективный и бесконфликтный способ распределения ресурсов ЭВМ между пользователями.

Границы, определяющие область действия понятия “ресурс”, достаточно условны. Поэтому будем полагать, что всякий потребляемый объект (независимо от формы его существования), обладающий некоторой практической ценностью для потребителя, является *ресурсом*.

Ресурсы различаются по запасу выделяемых единиц ресурса и бывают в этом смысле исчерпываемые и неисчерпываемые. Исчерпываемость ресурса, как правило, приводит к жизненным конфликтам в среде потребителей, Для регулирования конфликтов ресурсы должны распределяться между потребителями по каким-то правилам, в наибольшей степени их удовлетворяющим.

Вычислительную систему или ЭВМ можно представить как ограниченную последовательность функциональных элементов. обладающих потенциальными возможностями выполнения с их помощью или над ними действий, связанных с обработкой, хранением или передачей данных. Такие элементы “пользуются спросом”, т.е. потребляются другими элементами, являющимися в общем случае пользователями.

Уровень детализации элементов, выделяемых по запросам для использования в системе, может быть различным. Можно в качестве элемента, выделяемого для использования, рассматривать всю ЭВМ в целом. В равной степени отдельный разряд ячейки памяти можно трактовать как отдельный распределяемый элемент. Степень детализации зависит от того, кто и каким образом требует и использует выделяемые элементы системы. Поэтому для определенности будем считать, что потребителем выделяемых элементов вычислительной системы будут являться процессы, развивающиеся в ней. Все выделяемые по запросам от процессов элементы системы отождествляются с понятием ресурсов. Именно в этом смысле далее и трактуется понятие ресурса. В соответствии с ГОСТ 19781 - 83 *ресурсом является средство вычислительной системы, которое может быть выделено процессу на определенный интервал времени*.

Свойства и классификация ресурсов

Упорядоченность ресурсов по некоторым классификационным признакам нужна в ОС для определения тождественных действий в отношении ресурсов данного класса - действий по учету и распределению ресурсов, устранению конфликтов в их использовании и т.п.

Одним из важнейших свойств ресурса является "реальность существования". В этом смысле ресурсы разделяют на *физические* и *виртуальные* (мнимые). Под физическим понимают ресурс, который реально существует и при распределении его между пользователями обладает всеми присущими ему физическими характеристиками. Виртуальный ресурс схож многими своими характеристиками с некоторым физическим, но по многим свойствам и отличен. По сути - это некоторая модель физического ресурса. Виртуальный ресурс не существует в том виде, в котором он проявляет себя пользователю.

Свойство виртуализации ресурсов — одной из важнейших при построении систем управления ресурсами. По значимости — это одна из важнейших концепций при построении современных ОС.

Построение каждого виртуального ресурса проводится на базе некоторого физического. Имея всего один физический ресурс, можно построить на его основе несколько виртуальных. Это дает возможность существенно экономичнее использовать соответствующий физический ресурс, а также увеличивать гибкость политики распределения ресурсов, исключая в большинстве случаев конфликтные ситуации.

В зависимости от того, допускает ли физический ресурс виртуализацию, то есть построение на его основе виртуального ресурса, ресурсы можно разделить на *эластичные* и *жесткие*. Жесткий - физический ресурс, который по своим внутренним свойствам не допускает виртуализации.

В соответствии с признаком "степень активности" различают *активные* и *пассивные* ресурсы. Активный ресурс способен выполнять действия по отношению к другим ресурсам (или в отношении самого себя). ЦП - пример активного ресурса. Область памяти - пример пассивного. Очевидно, что логика распределения активных ресурсов должна отличаться от логики распределения пассивных.

Динамика ресурсов в отношении процессов позволяет выделить ресурсы *постоянные*, существующие до порождения процесса и на всем протяжении его существования и ресурсы *временные*, появляющиеся динамически в течение времени существования рассматриваемого процесса. Создание и уничтожение может производиться как самим процессом, так и другими процессами -системными или пользовательскими.

По "степени важности" можно выделить ресурсы *главные* и *второстепенные*. Главные - без которых данный процесс принципиально не может выполняться (ЦП, память). Второстепенные ресурсы допускают некоторое альтернативное развитие процесса при их отсутствии. Например процесс временно может обходиться без записи результатов на диск в случае неисправности последнего.

Действия над ресурсами

При централизованном распределении ресурсов соответствующими механизмами ОС в отношении каждого ресурса предполагается, что процесс-пользователь выполняет три типа действий: *запрос, использование,освобождение*. При выполнении действия *запрос* в ответ на требование процесса-пользователя система выделяет ресурс, либо отказывает в распределении. Отказ может быть вызван тем, что распределяемый ресурс находится в состоянии "Занят" либо обусловлен какой-то другой причиной. Если ресурс после выполнения действия *запрос* распределен процессу, то процесс может использовать его. Выполняется действие*использование*. Действие ОСВ *освобождение* выполняется по требованию процесса и сводится к переводу ресурса в состояние "Свободен".

Природа ресурса и (или) используемое правило распределения ресурса обусловлены параллельной или последовательной схемой использования распределяемого между несколькими процессами ресурса. Последовательная схема предполагает, что в отношении некоторого ресурса, который называют *последовательно используемым*, допустимо строго последовательное во времени выполнение цепочек действий *"запрос-исполнение-освобождение"* каждым процессом-потребителем этого ресурса. Для параллельных процессов такие цепочки действий являются *критическими областями* и должны выполняться так, чтобы удовлетворять правилу взаимного исключения, определенному ранее. Поэтому последовательно используемый ресурс, разделяемый несколькими параллельными процессами, чаще называют *критическим ресурсом*. В рассмотренном классе потребляемых ресурсов буфер, хранящий принятые, но еще не востребованные сообщения, является примером критического ресурса для процесса-производителя и процесса-потребителя соответственно.

Параллельная схема предполагает параллельное, т. е. одновременное, использование одного ресурса, который поэтому называют *параллельно используемым* более чем одним процессом. Такое использование не должно вносить каких-либо ошибок в логику развития каждого из процессов. Аналогично рассмотренному случаю, ресурс может быть параллельно используемым благодаря своей природе либо специальной организации действий при работе с ним. Массив данных, находящийся в некоторой области оперативной памяти и допускающий только чтение данных из него,— пример параллельно используемого ресурса.

Дисциплины распределения ресурсов

Использование многими процессами того или иного ресурса, который в каждый момент времени может обслуживать лишь один процесс, осуществляется с помощью дисциплин распределения ресурса. Их основой являются:

- дисциплины формирования очередей на ресурсы или совокупность правил, определяющих размещение процессов в очереди
- дисциплины обслуживания очереди или совокупность правил извлечения одного из процессов очереди с последующим представлением выбранному процессу ресурса для использования

Основным конструктивным, согласующим элементом при реализации той или иной дисциплины диспетчеризации является очередь, в которую по определенным правилам заносятся и извлекаются запросы.

Дисциплины формирования очередей разделяются на два класса:

1. статический, где приоритеты назначаются до выполнения пакета заданий
2. динамический, при котором приоритеты определяются в процессе выполнения пакета

Оба класса широко используются в практике организации вычислительного процесса в ЭВМ.

В ЭВМ не только многопрограммных, но и однопрограммных, однопроцессорных, многопроцессорных широко используется ряд дисциплин обслуживания очередей, ставших классическими. Эти дисциплины распределения ресурсов часто называют базовыми. Рассмотрим наиболее часто встречающиеся на практике дисциплины обслуживания.

Тема 12. Системные программы: утилиты, макроассемблеры, компиляторы, интерпретаторы, отладчики. Сохранность и защита программных систем

Системные программы: утилиты, макроассемблеры, компиляторы, интерпретаторы, отладчики

Утилиты – это программы, обеспечивающие выполнение вспомогательных функций при работе с компьютером, т.е. расширяющие возможности ОС компьютера. Можно выделить несколько групп утилит, которые наиболее полезны для различных компьютеров, в независимости от того, каким образом используются данный компьютер. К таким утилитам можно отнести утилиты:

- поиска неисправностей,
- деинсталляции,
- сжатия файлов,
- резервного копирования,
- просмотра файлов,
- утилиты для работы с Internet,
- антивирусные программы.

Некоторые утилиты встраиваются непосредственно в ОС.

Машинный код (также употребляются термины платформенно-ориентированный код, а также собственный код, родной код или нативный код — от англ. native code) — система команд (набор кодов операций) конкретной вычислительной машины, которая интерпретируется непосредственно процессором или микропрограммами данной вычислительной машины.

Каждая модель процессора имеет свой собственный набор команд, хотя во многих моделях эти наборы команд сильно перекрываются. Говорят, что процессор A совместим с процессором B, если процессор A полностью «понимает» машинный код процессора B. Если процессор A знает несколько команд, которых не понимает процессор B, то B несовместим с A.

Также инструкции бывают постоянной длины (у RISC-, MISC-архитектур) и диапазонной (у CISC-архитектур; например, для архитектуры x86 команда имеет длину от 8 до 120 битов).

Трансля́тор — программа или техническое средство, выполняющее трансляцию программы.[1][2]

Трансляция программы — преобразование программы, представленной на одном из языков программирования, в программу на другом языке и, в определённом смысле, равносильную первой.[1]

Транслятор обычно выполняет также диагностику ошибок, формирует словари идентификаторов, выдаёт для печати тексты программы и т. д.[1]

Язык, на котором представлена входная программа, называется исходным языком, а сама программа — исходным кодом. Выходной язык называется целевым языком или объектным кодом.

В общем случае понятие трансляции относится не только к языкам программирования, но и к другим языкам — как формальным компьютерным (вроде языков разметки типа HTML), так и естественным (русскому, английскому и т. п.).

Компиля́тор — программа или техническое средство, выполняющее компиляцию.

Компиляция — трансляция программы, составленной на исходном языке высокого уровня, в эквивалентную программу на низкоуровневом языке, близком машинному коду (абсолютный код, объектный модуль, иногда на язык ассемблера. Входной информацией для компилятора (исходный код) является описание алгоритма или программа на проблемно-ориентированном языке, а на выходе компилятора — эквивалентное описание алгоритма на машинно-ориентированном языке (объектный код).

Компилятор транслирует всю программу целиком и далее она отправляется на выполнение.

Интерпрета́тор — программа (разновидность транслятора) или аппаратное средство, выполняющее интерпретацию.

Интерпрета́ция — пооператорный (покомандный, построчный) анализ, обработка и тут же выполнение исходной программы или запроса (в отличие от компиляции, при которой программа транслируется без её выполнения).

Ассе́мблер (от англ. assembler — сборщик) — компьютерная программа, компилятор исходного текста программы, написанной на языке ассемблера, в программу на машинном языке.

Как и сам язык (ассемблер), ассемблеры, как правило, специфичны конкретной архитектуре, операционной системе и варианту синтаксиса языка. Вместе с тем существуют мультиплатформенные или вовсе универсальные (точнее, ограниченно-универсальные, потому что на языке низкого уровня нельзя написать аппаратно-независимые программы) ассемблеры, которые могут работать на разных платформах и операционных системах. Среди последних можно также выделить группу кросс-ассемблеров, способных собирать машинный код и исполняемые модули (файлы) для других архитектур и ОС.

Ассемблеры для DOS

Наиболее известными ассемблерами для операционной системы DOS являлись Borland Turbo Assembler (TASM), Microsoft Macro Assembler (MASM) и Watcom Assembler (WASM). Также в своё время был популярен простой ассемблер A86.

Windows

При появлении операционной системы Windows появилось расширение TASM, именуемое TASM 5+ (неофициальный пакет, созданный человеком с ником !tE), позволившее создавать программы для выполнения в среде Windows. Последняя известная версия TASM — 5.3, поддерживающая инструкции MMX, на данный момент включена в Turbo C++ Explorer. Но официально развитие программы полностью остановлено.

Microsoft поддерживает свой продукт под названием Microsoft Macro Assembler. Она продолжает развиваться и по сей день, последние версии включены в наборы DDK. Но версия программы, направленная на создание программ для DOS, не развивается. Кроме того, Стивен Хатчессон создал пакет для программирования на MASM под названием «MASM32».

GNU и GNU/Linux

В состав операционной системы GNU входит пакет binutils, включающий в себя ассемблер gas (GNU Assembler), использующий AT&T-синтаксис, в отличие от большинства других популярных ассемблеров, которые используют Intel-синтаксис (поддерживается с версии 2.10).

Переносимые ассемблеры

Также существует открытый проект ассемблера, версии которого доступны под различные операционные системы, и который позволяет получать объектные файлы для этих систем. Называется этот ассемблер NASM (Netwide Assembler).

Yasm — это переписанная с нуля версия NASM под лицензией BSD (с некоторыми исключениями).

flat assembler (fasm) — молодой ассемблер под модифицированной для запрета перелицензирования (в том числе под GNU GPL) BSD-лицензией. Есть версии для KolibriOS, Linux, DOS и Windows; использует Intel-синтаксис и поддерживает инструкции x86-64.

Макроассемблер (от греч. μάκρος — большой, обширный) — макропроцессор, базовым языком которого является язык ассемблера.

Макропроце́ссор (также макрогенера́тор) — программа, выполняющая преобразование входного текста в выходной при помощи задаваемых ей правил замены последовательностей символов, называемых правилами макроподстановки.

Наиболее простое и часто используемое правило макроподстановки сводится к замене определённой строки (называемой макро (макрос) или макрокоманда) другой строкой, возможно, с использованием параметров. Также правила макроподстановки могут иметь более сложный характер, включая определение процедур и функций, вычислительные алгоритмы и пр.

Макропроцессор, как таковой, является частным случаем транслятора. В то же время, некоторые макропроцессоры являются частью более сложных трансляторов — ассемблеров и компиляторов языков программирования. Широко распространено использование макропроцессоров при трансляции с языков ассемблера, в таком случае соответствующий ассемблер называется макроассемблером. Простейший макропроцессор является частью компилятора языка программирования Си. В компилятор языка программирования ПЛ/1 входит значительно более сложный макропроцессор, фактически сам являющийся подмножеством ПЛ/1. Макропроцессор, являющийся частью компилятора, называется препроцессор.

Также существуют самостоятельные макропроцессоры, такие как, например, макропроцессор m4.

К макропроцессорам может быть отнесено ядро системы компьютерной вёрстки TEX.

Граница между макропроцессорами, с одной стороны, и трансляторами языков программирования, ориентированных на символьную обработку, таких как REXX, Perl, Снобол, Рефал и пр., с другой — является достаточно условной. Обычно макропроцессорами называют такие трансляторы, входной язык которых малопригоден для написания универсальных программ, а ориентирован в основном на простые преобразования входного текста в выходной путём символьной подстановки.

Отладчик — это программный модуль, который позволяет выполнить основные задачи, связанные с мониторингом процесса выполнения результирующей прикладной программы. Этот процесс называется отладкой и включает в себя следующие основные возможности:

· последовательное пошаговое выполнение результирующей программы на основе шагов по машинным командам или по операторам входного языка;

· выполнение результирующей программы до достижения ею одной из заданных точек останова (адресов останова);

§ выполнение результирующей программы до наступления некоторых заданных условий, связанных с данными и адресами, обрабатываемыми этой программой;

§ просмотр содержимого областей памяти, занятых командами или данными результирующей программы.

Командный интерпретатор, интерпретатор командной строки - компьютерная программа, часть операционной системы, обеспечивающая базовые возможности управления компьютером посредством интерактивного ввода команд через интерфейс командной строки или последовательного исполнения пакетных командных файлов. Как правило его функции сводятся к предоставлению пользователю возможности запускать другие программы, может также содержать некоторые базовые команды ввода-вывода и свой простой скриптовый язык программирования. В операционные системы MS DOS и Windows 95 включен командный интерпретатор command.com, Windows NT включен cmd.exe, в OS/2 командный интерпретатор тоже называется cmd.exe, самый распространенный командный интерпретатор в Linux и FreeBSD — bash, помимо которого есть большое семейство других. Как правило, при низкоуровневой настройке ОС у пользователя есть возможность менять командный интерпретатор, используемый по умолчанию.

К функциям интерпретатора командной строки относятся:

- Взаимодействие с пользователем (редактирование командной строки, история команд и т.д.).
- Обработка (расширение) шаблонов имен ("*", "?" и т.д.).
- Перенаправление ввода-вывода команд.
- Управление заданиями.

Сохранность и защита программных систем

Чтобы понять, какого рода угрозам подвергаются компьютерные системы нужно определить требования к безопасности. Обычно выдвигаются такие четыре требования:

- Конфиденциальность. Согласно этому требованию, информацию от компьютерных систем могут получать только авторизованные лица. Доступ подобного рода включает в себя вывод на печать, на экран и другие формы предоставления информации, в том числе само обнаружение существования объекта.
- Целостность. Предполагает, что свойства компьютерной системы могут изменять только авторизованные лица. Под изменением подразумевается запись, редактирование, изменение статуса, удаление или создание новых объектов.
- Доступность. Необходимо, чтобы свойства компьютерной системы были доступны авторизованным лицам.
- Аутентичность. Компьютерная система должна иметь возможность проверять идентичность пользователя.

Чтобы понять, атаки какого вида представляют угрозу для компьютерной системы, рассмотрим ее работу в процессе предоставления информации. Вообще говоря, информация каким-либо образом переходит от источника (например, файла или области основной памяти) к получателю (например, другому файлу или пользователю). Можно выделить четыре общих категории атак:

- Прерывание. Компоненты системы выходят из строя, становятся недоступными или непригодными. Это атака, целью которой является нарушение доступности.
- Перехват. Это атака, целью которой является нарушение конфиденциальности, в результате чего доступ к компонентам системы получают несанкционированные стороны. В роли несанкционированной стороны может выступать лицо, программа или компьютер. В качестве примеров можно привести перехват передаваемых по сети сообщений или незаконное копирование файлов или программ.
- Изменение. Несанкционированная сторона не только получает доступ к системе, но и возможность нарушения целостности. В качестве примеров можно привести замену значений в файле данных, изменение программы таким образом, что она будет работать по-другому, а также изменение содержимого передаваемых по сети сообщений.

- Подделка. Несанкционированная сторона помещает в систему поддельные объекты. Целью этой атаки является нарушение аутентичности. В качестве примеров можно привести помещение в сеть поддельных сообщений или добавление записей в файл.

Компоненты компьютерной системы

Компоненты компьютерной системы можно разделить на следующие категории: аппаратное обеспечение, программное обеспечение, данные, а также линии связи и сети. Рассмотрим характер угроз, с которыми встречаются категории каждого вида.

Аппаратное обеспечение

Основная угроза для аппаратного обеспечения компьютерной системы связана с его доступностью. Аппаратное обеспечение больше всего подвержено атакам и менее всего поддается автоматическому управлению. В число угроз входят случайный и преднамеренный вывод оборудования из строя, а также его кража. Распространенность персональных компьютеров, рабочих станций и все более широкое использование локальных сетей приводят к увеличению потенциальной возможности потерь в этой области. Для устранения угрозы подобного рода нужны административные меры по предотвращению физического доступа к системам.

Программное обеспечение

Компьютерная система без программного обеспечения представляет собой бесполезный набор аппаратных устройств. Именно операционная система, утилиты и приложения делают компьютерную систему пригодной для использования отдельными лицами и организациями. Рассмотрим несколько различных угроз этой части компьютерной системы.

Основную опасность для программного обеспечения представляет атака на доступность. Программы, особенно прикладные, чрезвычайно легко удалить. Кроме того, программное обеспечение может быть изменено или повреждено, в результате чего оно станет непригодным для работы. Аккуратное управление настройкой конфигурации программ, включающее хранение резервных копий новейших версий, поможет повысить надежность их работы. Сложнее решить проблему, когда изменение программы приводит к тому, что она продолжает работать, но при этом ведет себя не так, как раньше. Эта категория атак связана с компьютерными вирусами.

Упомянем также о проблеме секретности программ. Несмотря на существование определенных мер защиты, в целом проблема несанкционированного копирования программного обеспечения остается неразрешенной.

Данные

Ответственность за безопасность аппаратного и программного обеспечения обычно возлагается на профессионалов вычислительного центра или на пользователей персонального компьютера. Безопасность данных является более обширной проблемой, включающей в себя безопасность файлов и других видов данных, управляемых отдельными лицами, группами и деловыми организациями.

Безопасность данных охватывает широкий круг вопросов, включающий в себя доступность, секретность и целостность. Когда речь идет о доступности, подразумевается защита от случайной либо преднамеренной порчи файлов с данными.

Очевидно, что для соблюдения секретности необходимо заботиться о предотвращении несанкционированного чтения файлов данных или баз данных. В этой области было проведено больше исследований и затрачено больше усилий, чем в любой другой области компьютерной безопасности. Не столь очевидной угрозой для секретности является анализ данных, заключающийся в использовании так называемых статистических баз данных, предоставляющих краткую или совокупную информацию. Совокупная информация обычно не представляет угрозы вмешательства в частную жизнь отдельных лиц. Однако в результате все более широкого использования статистических баз данных возрастает потенциальный риск раскрытия хранящейся в ней информации личного характера. По сути, в ходе тщательного анализа могут быть получены сведения об отдельных лицах, занесенных в базу данных. Это можно пояснить на таком простом примере: если в одной, таблице записан суммарный доход респондентов A, B, C и D, а в другой — суммарный доход респондентов A, B, C, D и E, то разность этих величин составит доход респондента E. Проблема обостряется в связи с комбинированием наборов данных. Во многих случаях в процессе составления необходимых совокупностей нужно извлекать отдельные составляющие, сопоставляя несколько различных наборов данных на уровнях агрегирования, подходящих для задачи.

Наконец, в большинстве систем основной задачей является сохранение целостности данных. Изменение файлов данных может иметь различные последствия — от незначительных до сокрушительных.

Линии связи и сети

Прослушивание или отслеживание передаваемой информации по своему характеру являются пассивными атаками. Цель атакующего состоит в получении этой информации. К данному виду атак относятся извлечение содержимого сообщения и анализ трафика.

Легко понять, зачем нужно *извлекать содержимое сообщения*. Во время телефонного разговора или с помощью электронной почты можно передать важную или конфиденциальную информацию. Нам не хотелось бы, чтобы оппонент узнал содержимое этих сообщений.

Другой вид пассивной атаки, *анализ трафика* (traffic analysis), является более сложным. Предположим, что у нас есть возможность скрыть содержимое передаваемой информации. Однако оппонент может получить представление о характере сообщений, несмотря на то что они зашифрованы. Он может определить местоположение и параметры узлов, обменивающихся информацией, а также собрать сведения о частоте передачи сообщений и об их размере. Полученные сведения могут дать представление о характере передаваемой информации.

Пассивные атаки очень трудно выявить, так как они не влекут за собой никаких изменений данных. Однако предотвратить эти атаки вполне возможно. Таким образом, внимание следует сосредоточить не на выявлении пассивных атак, а на их предотвращении.

Другой категорией атак являются активные атаки. Они Предполагают некоторое изменение потока данных или создание поддельного потока и подразделяются на четыре категории: имитация, воспроизведение, изменение сообщений и отказ от обслуживания.

Имитация имеет место, когда какой-то объект выдает себя за другой объект. Атака с имитацией обычно предпринимается вместе с активными атаками других видов. Например, может перехватываться, а затем воспроизводиться последовательность сообщений, передаваемых в процессе аутентификации, в результате чего авторизированные стороны с небольшими привилегиями получают дополнительные привилегии, выдавая себя за объект, обладающий ими.

Воспроизведение включает в себя пассивный перехват элементов данных с их последующей повторной передачей, чтобы произвести неавторизированный доступ.

Под *изменением, сообщений* подразумевается изменение какой-то части первоначального законного сообщения, удаление сообщений или изменение порядка их получения. Все это делается с целью получить несанкционированный доступ. Например, сообщение: "Позволить Ивану Сидорову читать конфиденциальные файлы" может быть изменено на такое: "Позволить Сидору Иванову читать конфиденциальные файлы".

Отказ от обслуживания препятствует нормальному использованию средств связи или управлению ими либо сдерживает их. Цель этой атаки может быть вполне конкретной; например, объект может подавлять все сообщения, предназначенные конкретному адресату (например, службе аудита безопасности). Другим видом отказа от обслуживания является подрыв работы всей сети, который достигается посредством вывода ее из строя либо перегрузки сообщениями в целях снижения ее производительности.

Активные атаки обладают характеристиками, противоположными характеристикам пассивных атак. Хотя пассивные атаки трудно выявить, их можно предотвратить с помощью специально разработанных мер. С другой стороны, активные атаки очень трудно полностью предотвратить, потому что для этого понадобилось бы обеспечить постоянную физическую защиту всех средств связи. Целесообразнее сосредоточить усилия на выявлении этих атак и устранении их последствий. Благодаря тому что выявление производит сдерживающий эффект, оно также может способствовать и предупреждению.

Защита

В основе многозадачности лежит способность системы предоставлять пользователям возможность совместного использования ресурсов. Объектом совместного использования является не только процессор, но и такие элементы, как:

• устройства ввода-вывода, например диски и принтеры;

• программы;

• данные.

Возможность совместного использования ресурсов предполагает их защиту. Операционная система может обеспечить разные степени защиты:

• Отсутствие защиты. Этот вариант подходит, если соответствующие процедуры выполняются во времени раздельно.

• Изоляция. При этом подходе каждый процесс работает отдельно от других процессов, не используя совместно с ними никаких ресурсов и не обмениваясь информацией. Каждый процесс имеет свое собственное адресное пространство, свои файлы и другие объекты.

• Полное разделение или полное его отсутствие. Владелец объекта (например, файла или сегмента памяти) объявляет его открытым либо закрытым. В первом случае доступ к объекту может получить любой процесс; во втором - доступ к этому объекту предоставляется только его владельцу.

• Совместное использование с ограничением доступа. Операционная система проверяет дозволенность доступа каждого отдельного пользователя к каждому отдельному объекту. В этом смысле

операционная система выступает в роли стража, гарантируя, что доступ к объектам получат только авторизованные пользователи.

• Совместное использование с помощью динамических возможностей. Этот вариант расширяет концепцию контроля доступа, позволяя динамически создавать права совместного использования объектов.

• Ограниченное использование объекта. При этой форме защиты ограничивается не столько доступ к объекту, сколько его использование. Например, пользователю может быть разрешено просматривать важный документ, но не распечатывать его. Приведем еще один пример: пользователь имеет доступ к базе данных, при котором он может извлекать статистические сводки, но не имеет возможности определять значения отдельных величин.

Выше перечислены различные варианты защиты в порядке возрастания сложности их реализации, что соответствует качеству предоставляемой защиты. Конкретная операционная система может предоставлять защиту разной степени для разных объектов, пользователей или приложений.

Требуется, чтобы в операционной системе поддерживался баланс между возможностями совместного использования компонентов компьютерной системы, способствующего повышению эффективности ее использования, и степенью защищенности ресурсов отдельных пользователей. Далее рассмотрены некоторые механизмы, с помощью которых в операционной системе обеспечивается защита.

Защита памяти

В многозадачной среде защита памяти становится важной проблемой. Здесь возникают вопросы, связанные не только с безопасностью, но и с правильной работой различных активных процессов. Если один из процессов неосторожно запишет что-то в область памяти другого процесса, то этим он может нарушить работу последнего.

Разделение пространства памяти между различными процессами легко осуществляется при использовании схемы виртуальной памяти. Если нужно обеспечить полную изоляцию, то операционной системе достаточно просто убедиться, что каждый сегмент или каждая страница доступна только тому процессу, которому она предоставлена. Этого легко добиться, потребовав, чтобы в таблицах страниц и/или сегментов не было повторяющихся элементов.

Если совместное использование разрешено, то один и тот же сегмент, или страница может появиться в нескольких таблицах. Этот тип совместного использования легче всего осуществить в системе, поддерживающей сегментацию или комбинацию сегментации с разбивкой на страницы. В среде с чисто страничной организацией памяти труднее провести черту между двумя областями памяти.

Контроль доступа, ориентированный на пользователя

Меры по контролю доступа, которые предпринимаются в системах обработки данных, можно разбить на две категории: имеющие отношение к пользователю и имеющие отношение к данным.

Контроль доступа, осуществляемый по отношению к пользователю, иногда неудачно называют аутентификацией. Мы воздержимся от использования этого термина, так как он широко применяется в связи с аутентификацией сообщений. Однако читателю следует иметь в виду, что в литературе он может употребляться в различном смысле.

Наиболее распространенным методом контроля доступа пользователя в совместно используемой системе или сервере является процедура регистрации, при которой пользователю нужно ввести свой идентификатор и пароль. Такая система совместного использования идентификатора и пароля зарекомендовала себя как не очень надежный метод контроля доступа пользователя. Пользователи иногда забывают свои пароли, случайно или непреднамеренно раскрывают их. Хакеры приобрели большие навыки разгадывания идентификаторов особых пользователей, осуществляющих управление системой, а также принадлежащих к персоналу, выполняющему ее обслуживание. Наконец, предпринимаются попытки проникнуть в файл, в котором хранятся идентификаторы и пароли.

Контроль доступа пользователей в распределенной среде может быть либо централизованным, либо децентрализованным. При централизованном подходе сеть обеспечивает сервис регистрации. При децентрализованном контроле доступа пользователей сеть рассматривается как прозрачный канал связи, а процедура регистрации пользователя выполняется на том узле, на который он пытается войти. Конечно же, в таком случае следует позаботиться о безопасной передаче паролей по сети.

Во многих сетях используется двухуровневый контроль доступа. Отдельные узлы могут быть оснащены регистрационными устройствами для защиты своих ресурсов и приложений, а вся сеть в целом может иметь защиту, ограничивающую доступ к ней несанкционированных пользователей, Такае двухуровневые средства желательно иметь в том случае, когда в одну сеть подключены неравноправные узлы, а сама она просто служит удобным средством доступа с терминала узла. В сети с более однородными узлами на центральном узле управления сетью можно обеспечить централизованную стратегию доступа.

Контроль доступа, ориентированный на данные

После успешной регистрации пользователю предоставляется доступ к одному или нескольким узлам и приложениям. Для системы, в базе данных которой содержится важная информация, этого в общем случае недостаточно. Пользователь может быть идентифицирован системой с помощью процедуры управления доступом. Каждому пользователю может соответствовать профиль, в котором указываются разрешенные операции и режимы доступа служат основой для определенных правил. Однако система управления базой данных должна управлять доступом к отдельным записям или даже к их частям.

Система управления файлами или базой данных основывается на общей модели под названием матрица доступа (access matrix). В эту модель входят такие понятия как:

• Субъект. Элемент, способный осуществлять доступ к объектам. Вообще говоря, понятие "субъект" приравнивается к понятию "процесс". Фактически любой пользователь или приложение получает доступ к объекту с помощью представляющего его процесса.

• Объект. Все то, доступ к чему контролируется. Примерами могут быть файлы, части файлов, программы и сегменты памяти.

• Право доступа. Способ, с помощью которого субъект осуществляет доступ к объекту. В качестве примеров можно привести чтение, запись и выполнение.

Одно измерение этой матрицы состоит из идентификаторов субъектов, которые могут попытаться получить доступ. Обычно в этот список входят идентификаторы отдельных пользователей или групп пользователей, хотя контроль доступа может производиться не в отношении пользователей, а в отношении терминалов, узлов или приложений. Второе измерение представляет собой список объектов, к которым может осуществляться доступ. На самом высоком уровне детализации в роли объектов могут выступать отдельные файлы данных. Объектами матрицы могут быть также более общие группы, такие, как записи, файлы или даже целые базы данных. В каждом элементе матрицы указаны права доступа данного субъекта к данному объекту.

4.3. Лабораторные работы

№ п/п	*Номер раздела дисциплины*	*Наименование лабораторной работы*	*Объем (час.)*	*Вид занятия в интерактивной, активной, инновационной формах, (час.)*
1	**2.**	Динамическое управление потоком работ в вычислительной системе	7	разбор конкретных ситуаций (3 час.)
2	**6.**	Диспетчеризация задач. Исследование и анализ алгоритмов	7	разбор конкретных ситуаций (2 час.)
3	**7.**	Управление виртуальной памятью	7	разбор конкретных ситуаций (2 час.)
4	**10, 11.**	Управление процессами. Моделирование функций многозадачной ОС	7	разбор конкретных ситуаций (3 час.)
5	**12.**	Разработка автомата распознавателя	6	разбор конкретных ситуаций (2 час.)
		ИТОГО	**34**	**12**

4.4. Практические занятия

Учебным планом не предусмотрено.

4.5. Контрольные мероприятия: курсовой проект (курсовая работа), контрольная работа, РГР, реферат

Учебным планом не предусмотрено.

5. МАТРИЦА СООТНЕСЕНИЯ РАЗДЕЛОВ УЧЕБНОЙ ДИСЦИПЛИНЫ К ФОРМИРУЕМЫМ В НИХ КОМПЕТЕНЦИЯМ И ОЦЕНКЕ РЕЗУЛЬТАТОВ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Компетенции *№, наименование разделов дисциплины*	*Кол-во часов*	*Компетенции*		*Σ комп.*	$t_{ср}$, *час*	*Вид учебных занятий*	*Оценка результатов*
		ПК-2	*ОПК-7*				
1	2	3	4	5	6	7	8
1.Функции и организация операционных систем (ОС). Обзор современных ОС	7,5	+	+	2	3,75	Лк, СР	Зачет
2.Процессы. Операции над процессами. Процессы и нити. Идентификация и группирование процессов. Классификация процессов и ресурсов	13,5	+	+	2	6,75	Лк, ЛР,СР	Зачет
3.Задачи синхронизации. Семафорная техника синхронизации	6,5	+	+	2	3,25	Лк, СР	Зачет
4.Тупики. Условия возникновения, предупреждение и обходы	6,5	+	+	2	3,25	Лк, СР	Зачет
5.Межпроцессорные коммуникации (сигнальный механизм, очереди сообщений, разделяемые сегменты памяти, сокеты)	6,5	+	+	2	3,25	Лк, СР	Зачет
6.Системные часы и таймеры. Планирование выполнения процессов. Диспетчеризация процессов реального времени	13,5	+	+	2	6,75	Лк, ЛР,СР	Зачет
7.Организация и управление памятью	13,5	+	+	2	6,75	Лк, ЛР,СР	Зачет
8.Файловая система. Управление вводом/выводом. Варианты структур ядра ОС	6,5	+	+	2	3,25	Лк, СР	Зачет
9.Мультипроцессорные ОС, сетевые ОС, распределенные ОС: назначение и подходы к построению. Особенности сетевых ОС	5,5	+	+	2	2,75	Лк, СР	Зачет
10.Вычислительный процесс. Обслуживание прерываний	9,5	+	+	2	4,75	Лк, ЛР,СР	Зачет
11.Многозадачные и многопользовательские ОС. Распределение ресурсов в ОС.	8	+	+	2	4	Лк, ЛР,СР	Зачет
12.Системные программы: утилиты, макроассемблеры, компиляторы, интерпретаторы, отладчики. Сохранность и защита программных систем	11	+	+	2	5,5	Лк, ЛР,СР	Зачет
всего часов	**108**	**99**	**99**	**2**	**108**		

6. ПЕРЕЧЕНЬ УЧЕБНО-МЕТОДИЧЕСКОГО ОБЕСПЕЧЕНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ

Молчанов А. Ю. Системное программное обеспечение : учебник для вузов /. - Санкт-Петербург : Питер, 2011. - 399 с. - Учебник для вузов., стр.55-320.

7. ПЕРЕЧЕНЬ ОСНОВНОЙ И ДОПОЛНИТЕЛЬНОЙ ЛИТЕРАТУРЫ, НЕОБХОДИМОЙ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

№	*Наименование издания*	*Вид занятия*	*Количество экземпляров в библиотеке,* **шт.**	*Обеспеченность, (экз./ чел.)*
1	2	3	4	5
Основная литература				
1.	Молчанов А.Ю.Системное программное обеспечение : учебник для вузов /. - Санкт-Петербург : Питер, 2011. - 399 с. - Учебник для вузов	Лк, ЛР	12	1
Дополнительная литература				
2.	Коликова Т.В. Операционные системы. Учебное пособие. Санкт-Петербург, 2012 г. – 214с. http://elib.spbstu.ru/dl/2688.pdf/view	Лк, ЛР	ЭР	1

8. ПЕРЕЧЕНЬ РЕСУРСОВ ИНФОРМАЦИОННО - ТЕЛЕКОММУНИКАЦИОННОЙ СЕТИ «ИНТЕРНЕТ» НЕОБХОДИМЫХ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

1. Электронный каталог библиотеки БрГУ
http://irbis.brstu.ru/CGI/irbis64r_15/cgiirbis_64.exe?LNG=&C21COM=F&I21DBN=BOOK&P21DBN=BOOK&S21CNR=&Z21ID=.
2. Электронная библиотека БрГУ
http://ecat.brstu.ru/catalog .
3. Электронно-библиотечная система «Университетская библиотека online»
http://biblioclub.ru .
4. Электронно-библиотечная система «Издательство «Лань»
http://e.lanbook.com .
5. Информационная система "Единое окно доступа к образовательным ресурсам"
http://window.edu.ru .
6. Научная электронная библиотека eLIBRARY.RU http://elibrary.ru .
7. Университетская информационная система РОССИЯ (УИС РОССИЯ)
https://uisrussia.msu.ru/ .
8. Национальная электронная библиотека НЭБ
http://xn--90ax2c.xn--p1ai/how-to-search /.

9. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

9.1. Методические указания для обучающихся по выполнению лабораторных работ

Лабораторная работа № 1

Динамическое управление потоком работ в вычислительной системе.

Цель работы:

Изучить информацию о процессах и потоках Windows NT и написать программу.

Вид занятия в интерактивной, активной форме: Используя функцию ***CreateProcess***, создайте процесс для запуска notepad.exe.

Задание:

Изучить информацию о процессах и потоках Windows NT.

Порядок выполнения:

Написать программы на С, используя функции ***CreateProcess*** и ***CreateThread*** для создания:

- одного или нескольких процессов (каждый с базовым потоком)**;**
- нескольких потоков в одном процессе.

Форма отчетности:

Отчет по лабораторной работе, скрепленный титульным листом. Отчет должен содержать название работы, цель, задание и результат выполнения задания.

Задания для самостоятельной работы:

Изучить информацию о процессах и потоках Windows NT .

Рекомендации по выполнению заданий и подготовке к практическому занятию

Ознакомиться с теоретическим материалом, представленным во втором разделе данной дисциплины и учебном пособии.

Основная литература

1. А. Ю. Молчанов Системное программное обеспечение : учебник для вузов /. - Санкт-Петербург : Питер, 2011. - 399 с. - Учебник для вузов.

Дополнительная литература

1. Т.В. Коликова. Операционные системы. Учебное пособие. Санкт-Петербург, 2012 г. http://elib.spbstu.ru/dl/2688.pdf/view

Контрольные вопросы для самопроверки

1. Что такое поток?
2. Что такое процесс?
3. Назвать основные приоритеты процессов?

Лабораторная работа № 2

Диспетчеризация задач. Исследование и анализ алгоритмов.

Цель работы:

Изучить средства для анализа процессов и потоков средствами Windows.

Вид занятия в интерактивной, активной форме: Ознакомиться с программами winmsd.exe, Task Manager, pview.exe.

Задание:

Изучить средства для анализа процессов и потоков средствами Windows.

Порядок выполнения:

Изучить информацию о процессах и потоках программами winmsd.exe, Task Manager, pview.exe.

Форма отчетности:

Отчет по лабораторной работе, скрепленный титульным листом. Отчет должен содержать название работы, цель, задание и результат выполнения задания.

Изучить информацию о процессах и потоках Windows средствами системы.

Рекомендации по выполнению заданий и подготовке к практическому занятию

Ознакомиться с теоретическим материалом, представленным в шестом разделе данной дисциплины и учебном пособии.

Основная литература

1. А. Ю. Молчанов Системное программное обеспечение : учебник для вузов /. - Санкт-Петербург : Питер, 2011. - 399 с. - Учебник для вузов.

Дополнительная литература

1. Т.В. Коликова. Операционные системы. Учебное пособие. Санкт-Петербург, 2012 г. http://elib.spbstu.ru/dl/2688.pdf/view

Контрольные вопросы для самопроверки

1.Назовите системные процессы.

2. Назовите пользовательские процессы.

Лабораторная работа № 3

Управление виртуальной памятью

Цель работы:

Разработать программу управления управление виртуальной памятью.

Вид занятия в интерактивной, активной форме: выполнить задание и ознакомиться с функциями управления виртуальной памятью.

Задание:

В соответствии с заданием, полученным от преподавателя разработать программу.

Порядок выполнения:

Изучить функции.

Составить программу управления.

Форма отчетности:

Отчет по лабораторной работе, скрепленный титульным листом. Отчет должен содержать название работы, цель, задание и результат выполнения задания.

Задания для самостоятельной работы:

В соответствии с заданием, полученным от преподавателя разработать программу.

Рекомендации по выполнению заданий и подготовке к практическому занятию

Ознакомиться с теоретическим материалом, представленным в седьмом разделе данной дисциплины и учебном пособии.

Основная литература

1. А. Ю. Молчанов Системное программное обеспечение : учебник для вузов /. - Санкт-Петербург : Питер, 2011. - 399 с. - Учебник для вузов.

Дополнительная литература

1. Т.В. Коликова. Операционные системы. Учебное пособие. Санкт-Петербург, 2012 г. http://elib.spbstu.ru/dl/2688.pdf/view

Контрольные вопросы для самопроверки

1.Назвать функции для работы по управлению виртуальной памятью.

2. Назвать функции для работы по мониторингу виртуальной памяти.

Лабораторная работа № 4

Управление процессами. Моделирование функций многозадачной ОС

Цель работы:

Изучить синхронизацию родственных процессов с использованием наследования описателей и реализовать ее программно.

Вид занятия в интерактивной, активной форме: Использовать любые синхронизационные объекты для синхронизации нескольких (например, двух) потоков в одном процессе. В качестве разделяемого ресурса использовать стандартный ввод/вывод.

Задание:

1. Синхронизировать работу потоков процесса.

Порядок выполнения:

На основании представленного преподавателем задания синхронизации нескольких (например, двух) потоков в одном процессе.

Форма отчетности:

Отчет по лабораторной работе, скрепленный титульным листом. Отчет должен содержать название работы, цель, задание и результат выполнения задания.

Задания для самостоятельной работы:

В соответствии с заданием, полученным от преподавателя разработать программу.

Рекомендации по выполнению заданий и подготовке к практическому занятию

Ознакомиться с теоретическим материалом, представленным в 10 и 11 разделе данной дисциплины и учебном пособии.

Основная литература

1. А. Ю. Молчанов Системное программное обеспечение : учебник для вузов /. - Санкт-Петербург : Питер, 2011. - 399 с. - Учебник для вузов.

Дополнительная литература

1. Т.В. Коликова. Операционные системы. Учебное пособие. Санкт-Петербург, 2012 г. http://elib.spbstu.ru/dl/2688.pdf/view

Контрольные вопросы для самопроверки

1.Что такое синхронизация?

2.Назвать основные функции работы с процессами

Лабораторная работа № 5

Разработка автомата распознавателя

Цель работы:

Изучение основных понятий теории грамматик простого и операторного предшествования, ознакомление с алгоритмами синтаксического анализа (разбора) для некоторых классов КС-грамматик, получение практических навыков создания простейшего синтаксического анализатора для заданной грамматики операторного предшествования.

Вид занятия в интерактивной, активной форме: написать программу, которая выполняет лексический анализ входного текста в соответствии с заданием, порождает таблицу лексем и выполняет синтаксический разбор текста по заданной грамматике с построением дерева разбора.

Задание:

Получить задание от преподавателя по разработке программы.

Порядок выполнения:

На основании представленной программы управления робота изучить функциональные возможности и режимы работы.

Форма отчетности:

Отчет по лабораторной работе, скрепленный титульным листом. Отчет должен содержать название работы, цель, задание и результат выполнения задания.

Задания для самостоятельной работы:

Изучить основы теории грамматик простого и операторного предшествования.

Рекомендации по выполнению заданий и подготовке к практическому занятию

Ознакомиться с теоретическим материалом, представленным в 12-м разделе данной дисциплины и учебном пособии.

Основная литература

1. А. Ю. Молчанов Системное программное обеспечение : учебник для вузов /. - Санкт-Петербург : Питер, 2011. - 399 с. - Учебник для вузов.

Дополнительная литература

1. Т.В. Коликова. Операционные системы. Учебное пособие. Санкт-Петербург, 2012 г. http://elib.spbstu.ru/dl/2688.pdf/view

Контрольные вопросы для самопроверки

1.Что такое грамматика?
2.Что такое лексема?
3.Назвать функции анализаторов?

10. ПЕРЕЧЕНЬ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, ИСПОЛЬЗУЕМЫХ ПРИ ОСУЩЕСТВЛЕНИИ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ

Информационно-коммуникационные технологии (ИКТ) – преподаватель использует для:

- получения информации при подготовке к занятиям,
- создания презентационного сопровождения лекций;
- интерактивного общения;
- ОС Windows 7 Professional;
- Microsoft Office 2007 Russian Academic OPEN NO Level;
- Антивирусное программное обеспечение Kaspersky Security;
- ПО “Антиплагиат”

11. ОПИСАНИЕ МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЙ БАЗЫ, НЕОБХОДИМОЙ ДЛЯ ОСУЩЕСТВЛЕНИЯ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ

Вид занятия	*Наименование аудитории*	*Перечень основного оборудования*	*№ ЛР или ПЗ*
1	2	3	4
ЛР	Дисплейные классы	Персональные компьютеры	ЛР № 1-5
СР	ЧЗЗ	-	-

Приложение 1

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ПРОВЕДЕНИЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ

1. Описание фонда оценочных средств (паспорт)

№ компетенции	Элемент компетенции	Раздел	Тема	ФОС
ОПК-7	способность учитывать современные тенденции развития электроники, измерительной и вычислительной техники, информационных технологий в своей профессиональной деятельности	**1.**Функции и организация операционных систем (ОС). Обзор современных ОС	1.1 Системное программное обеспечение. 1.2 Операционная система 1.3 Обзор современных ОС	Вопросы к зачету
		2.Процессы. Операции над процессами. Процессы и нити. Идентификация и группирование процессов. Классификация процессов и ресурсов	2.1 Процессы и нити. 2.2 Операции над процессами. 2.3 Классификация процессов и ресурсов	Вопросы к зачету
		3.Задачи синхронизации. Семафорная техника синхронизации	3.1 Задачи синхронизации 3.2 Семафорная техника синхронизации 3.3 Проблемы синхронизации	Вопросы к зачету
		4.Тупики. Условия возникновения, предупреждение и обходы	4.1 Тупики 4.1Условия возникновения тупиков 4.3Предупреждение и обходы тупиков	Вопросы к зачету
		5.Межпроцессорные коммуникации (сигнальный механизм, очереди сообщений, разделяемые сегменты памяти, сокеты)	5.1 Сигнальный механизм, очереди сообщений 5.2 Разделяемые сегменты памяти 5.3 Сокеты	Вопросы к зачету
ПК-2	способность проводить вычислительные эксперименты с использованием стандартных программных средств с целью получения математических моделей	**6.**Системные часы и таймеры. Планирование выполнения процессов. Диспетчеризация процессов реального времени	6.1 Системные часы и таймеры 6.2 Планирование выполнения процессов 6.3 Диспетчеризация процессов реального времени	Вопросы к зачету
		7.Организация и управление памятью	7.1 Страничное распределение 7.2 Сегментное распределение 7.3 Сегментно-страничное распределение	Вопросы к зачету

		8.Файловая система. Управление вводом/выводом. Варианты структур ядра ОС	8.1 Файловая система 8.2 Управление вводом/выводом. 8.3 Варианты структур ядра ОС	Вопросы к зачету
		9.Мультипроцессорные ОС, сетевые ОС, распределенные ОС.	9.1 Мультипроцессорные ОС 9.2 Сетевые ОС 9.3 Распределенные ОС	Вопросы к зачету
		10.Вычислительный процесс. Обслуживание прерываний	10.1 Система прерывания 10.2 Классификация типов прерывания 10.3 Основные характеристики систем прерывания	Вопросы к зачету
		11.Многозадачные и многопользовательские ОС. Распределение ресурсов в ОС.	11.1 Многозадачные и многопользовательские ОС 11.2 Распределение ресурсов в ОС	Вопросы к зачету
		12.Системные программы: утилиты, макроассемблеры, компиляторы, интерпретаторы, отладчики. Сохранность и защита программных систем	12.1 Системные программы: утилиты, макроассемблеры, компиляторы, интерпретаторы, отладчики. 12.2 Сохранность и защита программных систем	Вопросы к зачету

2. Вопросы к зачету

№ п/п	Компетенции		ВОПРОСЫ К ЗАЧЕТУ	№ и наименование раздела
	Код	Определение		
1	2	3	4	5
1.	ОПК-7	способность учитывать современные тенденции развития электроники, измерительной и вычислительной техники, информационных технологий в своей профессиональной деятельности	1.1 Системное программное обеспечение. 1.2 Операционная система 1.3 Обзор современных ОС 2.1 Процессы и нити. 2.2 Операции над процессами. 2.3 Классификация процессов и ресурсов	**1**.Функции и организация операционных систем (ОС). Обзор современных ОС **2**.Процессы. Операции над процессами. Процессы и нити. Идентификация и группирование процессов. Классификация процессов и ресурсов

			3.1 Задачи синхронизации 3.2 Семафорная техника синхронизации 3.3 Проблемы синхронизации	**3**.Задачи синхронизации. Семафорная техника синхронизации
			4.1 Тупики 4.1Условия возникновения тупиков 4.3Предупреждение и обходы тупиков	**4**.Тупики. Условия возникновения, предупреждение и обходы
			5.1 Сигнальный механизм, очереди сообщений 5.2 Разделяемые сегменты памяти 5.3 Сокеты	**5**.Межпроцессорные коммуникации (сигнальный механизм, очереди сообщений, разделяемые сегменты памяти, сокеты)
2.	ПК-2	способность проводить вычислительные эксперименты с использованием стандартных программных средств с целью получения математических моделей	6.1 Системные часы и таймеры 6.2 Планирование выполнения процессов 6.3 Диспетчеризация процессов реального времени	**6**.Системные часы и таймеры. Планирование выполнения процессов. Диспетчеризация процессов реального времени
			7.1 Страничное распределение 7.2 Сегментное распределение 7.3 Сегментно-страничное распределение	**7**.Организация и управление памятью
			8.1 Файловая система 8.2 Управление вводом/выводом. 8.3 Варианты структур ядра ОС	**8**.Файловая система. Управление вводом/выводом. Варианты структур ядра ОС
			9.1 Мультипроцессорные ОС 9.2 Сетевые ОС 9.3 Распределенные ОС	**9**.Мультипроцессорные ОС, сетевые ОС, распределенные ОС: назначение и подходы к построению. Особенности

				сетевых ОС
			10.1 Система прерывания 10.2 Классификация типов прерывания 10.3 Основные характеристики систем прерывания	**10**.Вычислительный процесс. Обслуживание прерываний
			11.1 Многозадачные и многопользовательские ОС 11.2 Распределение ресурсов в ОС	**11**.Многозадачные и многопользовательские ОС. Распределение ресурсов в ОС.
			12.1 Системные программы: утилиты, макроассемблеры, компиляторы, интерпретаторы, отладчики. 12.2 Сохранность и защита программных систем	**12**.Системные программы: утилиты, макроассемблеры, компиляторы, интерпретаторы, отладчики. Сохранность и защита программных систем

3. Описание показателей и критериев оценивания компетенций

Показатели	Оценка	Критерии
Знать ОПК-7: – принципы организации операционных систем; – классификацию процессов и ресурсов ПК-2: – организацию и управление памятью **Уметь** ОПК-7: - управлять настройками файловой системы и управлением вводом/выводом ПК-2: - настраивать оборудование для выполнения заданных алгоритмов работы **Владеть** ОПК-7: - навыками настройки, программирования и управления системными ресурсами вычислительных машин ПК-2: - навыками вычислительных экспериментов с использованием стандартных программных средств	**зачтено**	Оценка «зачтено» выставляется в случае, если студент демонстрирует: – всестороннее систематическое знание программного материала; – правильное выполнение практических заданий, направленных на применение программного материала; – правильное применение основных положений программного материала.
	незачтено	Оценка «незачтено» выставляется в случае, если студент демонстрирует: – существенные пробелы в знании программного материала; – принципиальные ошибки при выполнении лабараторных заданий, направленных на применение программного материала; – невозможность применения основных положений программного материала.

4. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и опыта деятельности

Дисциплина Системное программное обеспечение направлена на изучение основ системных ресурсов ОС, формирование навыков использования системных ресурсов и разработки системного программного обеспечения для решения задач управления, методов контроля, обработки, анализа теоретических и экспериментальных исследований в сфере профессиональной деятельности.

Изучение дисциплины предусматривает:

- лекции,
- лабораторные занятия,
- самостоятельную работу,
- зачет

В ходе освоения раздела 1 «Функции и организация операционных систем (ОС). Обзор современных ОС» обучающиеся должны изучить основные понятия, определения, термины существующие в дисциплине, изучить состав современных ОС.

В ходе освоения раздела 2 «Процессы. Операции над процессами. Процессы и нити. Идентификация и группирование процессов. Классификация процессов и ресурсов» обучающиеся должны знать операции над процессами, идентификацию и группирование процессов.

В ходе освоения раздела 3 «Задачи синхронизации. Семафорная техника синхронизации» обучающиеся должны знать задачи синхронизации и семафоров.

В ходе освоения раздела 4 «Тупики. Условия возникновения, предупреждение и обходы» обучающиеся должны знать основные условия возникновения, предупреждение и обходы тупиков.

В ходе освоения раздела 5 «Межпроцессорные коммуникации (сигнальный механизм, очереди сообщений, разделяемые сегменты памяти, сокеты)» обучающиеся должны знать механизм системных сообщений.

В ходе освоения раздела 6 «Системные часы и таймеры. Планирование выполнения процессов. Диспетчеризация процессов реального времени» обучающиеся должны знать систему диспетчеризации процессов.

В ходе освоения раздела 7 «Организация и управление памятью» обучающиеся должны знать основы организации и управления памятью.

В ходе освоения раздела 8 «Файловая система. Управление вводом/выводом. Варианты структур ядра ОС» обучающиеся должны знать основный функции файловых систем и управления вводом/выводом.

В ходе освоения раздела 9 «Мультипроцессорные ОС, сетевые ОС, распределенные ОС: назначение и подходы к построению. Особенности сетевых ОС» обучающиеся должны знать классификацию ОС и их особенности.

В ходе освоения раздела 10 «Вычислительный процесс. Обслуживание прерываний» обучающиеся должны знать систему обслуживания прерываний.

В ходе освоения раздела 11 «Многозадачные и многопользовательские ОС. Распределение ресурсов в ОС» обучающиеся должны знать многозадачные и многопользовательские ОС и распределение ресурсов в ОС.

В ходе освоения раздела 12 «Системные программы: утилиты, макроассемблеры, компиляторы, интерпретаторы, отладчики. Сохранность и защита программных систем» обучающиеся должны знать основные системные программы: утилиты, макроассемблеры, компиляторы, интерпретаторы, отладчики.

В процессе выполнения лабораторных работ происходит закрепление знаний, формирование умений и навыков управления процессами и потоками, памятью, системными ресурсами ОС вычислительной системы.

Работа с литературой является важнейшим элементом в получении знаний по дисциплине. Прежде всего, необходимо воспользоваться списком рекомендуемой по данной дисциплине литературой. Дополнительные сведения по изучаемым темам можно найти в периоди-

ческой печати и Интернете.

К зачету допускаются студенты, которые выполнили и оформили все лабораторные работы.

Оценка знаний, умений, навыков осуществляется в процессе промежуточной аттестации обучающихся по дисциплине, которая осуществляется в виде зачета. Для оценивания знаний, умений, навыков используются ФОС по дисциплине, содержащие вопросы к зачету.

Приложение 2

АННОТАЦИЯ
рабочей программы дисциплины
Системное программное обеспечение

1. Цель и задачи дисциплины

Целью изучения дисциплины является: приобретение умений и навыков исследования проблем в своей предметной области, выбора методов и средств их решения, анализа результатов теоретических и экспериментальных исследований.

Задачей изучения дисциплины является: формирование способностей анализа результатов исследований, выбора методов и средств решения проблем в своей предметной области.

2. Структура дисциплины

2.1 Распределение трудоемкости по отдельным видам учебных занятий, включая самостоятельную работу: лекции- 17ч., лабораторные занятия-34ч., самостоятельная работа-57ч.

Общая трудоемкость дисциплины составляет 108 часа, 3 зачетных единицы.

2.2 Основные разделы дисциплины:

1.Функции и организация операционных систем (ОС). Обзор современных ОС
2.Процессы. Операции над процессами. Процессы и нити. Идентификация и группирование процессов. Классификация процессов и ресурсов
3.Задачи синхронизации. Семафорная техника синхронизации
4.Тупики. Условия возникновения, предупреждение и обходы
5.Межпроцессорные коммуникации (сигнальный механизм, очереди сообщений, разделяемые сегменты памяти, сокеты)
6.Системные часы и таймеры. Планирование выполнения процессов. Диспетчеризация процессов реального времени
7.Организация и управление памятью
8.Файловая система. Управление вводом/выводом. Варианты структур ядра ОС
9.Мультипроцессорные ОС. Сетевые ОС. Распределенные ОС.
10.Вычислительный процесс. Обслуживание прерываний
11.Многозадачные и многопользовательские ОС. Распределение ресурсов в ОС.
12.Системные программы: утилиты, макроассемблеры, компиляторы, интерпретаторы, отладчики. Сохранность и защита программных систем

3. Планируемые результаты обучения (перечень компетенций)

Процесс изучения дисциплины направлен на формирование следующих компетенций:
-ОПК-7: способность учитывать современные тенденции развития электроники, измерительной и вычислительной техники, информационных технологий в своей профессиональной деятельности.
-ПК-2: способность проводить вычислительные эксперименты с использованием стандартных программных средств с целью получения математических моделей

4. Вид промежуточной аттестации: зачет

Приложение 3

Протокол о дополнениях и изменениях в рабочей программе
на 20___-20___ учебный год

1. В рабочую программу по дисциплине вносятся следующие дополнения:
_____
_____

2. В рабочую программу по дисциплине вносятся следующие изменения:

_____
_____
_____

Протокол заседания кафедры №____ от «___» _____ 20 __ __ г.,
(разработчик)

Заведующий кафедрой _____ _____
(подпись) *(Ф.И.О.)*