Государственный Комитет Связи, Информатизации и Телекомуникационных Технологий Республики Узбекистан
Ташкентский Университет Информационных Технологий

КУРСОВАЯ РАБОТА

по курсу «Системное Программное Обеспечение»

на тему:

«Организация вычислительных процессов. Диспетчер задач»

Выполнил: Гуломов Т.Р.
Группа: 225-10 ИТр
Проверила: Акбарова М.Х.

Ташкент - 2013

Оглавление

Введение в предметную область 3
Требования к современным ОС 3
Теоретические основы диспетчеризации 3
Стратегии планирования 4
Дисциплины диспетчеризации 4
Виды дисциплин обслуживания: 5
Качество диспетчеризации и гарантии обслуживания 5
Программная модель "Диспетчеризация задач". 5
Исходные данные 7
Раздел №1 7
1. Временная диаграмма мультипрограммной работы ЭВМ (ДО FIFO) 7
2. Временная диаграмма для алгоритма FIFO 9
3. Временная диаграмма мультипрограммной работы ЭВМ (ДО SJF) 9
4. Временная диаграмма для алгоритма SJF 11
5. Выводы 12
Раздел №2 12
1. Задание 12
2. Исходные условия 12
3. Диспетчеризация задач для бесприоритетной ДО LIFO 12
4. Схема диспетчера бесприоритетной ДО LIFO 13
5. Диспетчеризация задач для приоритетной ДО с динамическим приоритетом 13
6. Схема диспетчера приоритетной ДО с динамическим приоритетом 13
7. Окончание работы диспетчеров 13
8. Алгоритм функционирования диспетчера 14
9. Текст программы диспетчера 15
Список литературы 18

Введение в предметную область

Современные вычислительные системы состоят из процессоров, памяти, таймеров, дисков, накопителей на магнитных лентах, сетевой коммуникационной аппаратуры, принтеров и других устройств. Функцией операционной системы (ОС) является распределение процессора, памяти, устройств и данных между процессами, конкурирующими за эти ресурсы. ОС должна управлять всеми ресурсами вычислительной машины таким образом, чтобы обеспечить максимальную эффективность её функционирования. Критерием эффективности может быть, например, пропускная способность или реактивность системы. Управление ресурсами включает решение двух общих, не зависящих от типа ресурса задач;

- ✓ Планирование ресурса, т.е. определение, кому, когда, а для делимых ресурсов и в каком количестве, необходимо выделить данный ресурс;
- ✓ Отслеживание состояния ресурса, т.е. поддержание оперативной информации о том, занят или не занят ресурс, а для делимых ресурсов – какое количество ресурса уже распределено, а какое свободно.

Для решения этих общих задач управления ресурсами разные ОС используют различные алгоритмы, что в конечном счете и определяет их облик в целом, включая характеристики производительности, область применения и даже пользовательский интерфейс. Так, например, алгоритм управления процессором в значительной степени определяет, является ли ОС системой разделения времени, системой пакетной обработки или системой реального времени.

В дисциплины “Системное программное обеспечение” студент должен изучить следующие обязательные разделы: «Назначение, функции и структура операционной системы (ОС); понятие процесса; управление процессами, способы диспетчеризации процессов; понятие ресурса, виды ресурсов, управление ресурсами; управление памятью; устройства, виды устройств, драйверы устройств, устройства в MS-DOS; файловая система на диске, структура логического диска в MS-DOS; синхронизация процессов, семафоры, сообщения, использование семафоров для решения задач взаимоисключения и синхронизации; тупики, способы борьбы с тупиками; и т.д.”.

В технической литературе термин «системное программное обеспечение» означает программы и комплексы программ, являющиеся общими для всех, кто совместно использует технические средства компьютера, и применяемые как для автоматизации разработки (создания) новых программ, так и для организации выполнения программ существующих. С этих позиций системное программное обеспечение может быть разделено на следующие пять групп:

1. *Операционные системы* – комплекс взаимосвязанных программ, предназначенный для повышения эффективности аппаратуры компьютера путём рационального управления его ресурсами, а также для обеспечения удобств пользователю путём предоставления ему расширенной виртупльной машины.
1. *Системы управления файлами.*
2. *Интерфейсные оболочки для взаимодействия пользователя с ОС и программные среды.*
3. *Системы программирования.*
4. *Утилиты.*

Требования к современным ОС

- Расширяемость
- Переносимость
- Совместимость
- Надёжность и отказоустойчивость
- Безопастность
- Производительность

Теоретические основы диспетчеризации

Планирование распределения процессора производится на нескольких уровнях. Один из них - средний уровень планирования - диспетчеризация. На этом уровне диспетчер задач (планировщик процессов) выбирает одну из задач из числа готовых к выполнению и предоставляет ей процессор.

Каждая задача занимает процессор относительно малое время (как правило, недостаточное для выполнения задачи), затем диспетчирование повторяется, процессор выделяется другой задаче. Диспетчер принимает текущие решения в динамике сложившейся конкретной обстановки.

Таким образом, цели диспетчирования задач следующие:

- распределение центрального процессора в динамике в соответствии с критериями;
- эффективная отработка алгоритмов управления задачами.

Итак: диспетчер - это программа, которая выбирает задачи (процессы) из "очереди на выполнение", переводит их в активное состояние и передает им контроль над центральным процессором.

Возникает задача подбора такого множества процессов, что при выполнении они будут как можно реже конфликтовать из-за имеющихся в системе ресурсов. Такая задача называется планированием вычислительных процессов.

На первый план уже очень давно вышли задачи динамического (или краткосрочного) планирования, то есть текущего наиболее эффективного распределения ресурсов, возникающего практически при каждом событии. Задачи динамического планирования стали называть диспетчеризацией.

Стратегии планирования

Прежде всего следует отметить, что при рассмотрении стратегий планирования, как правило, идет речь о краткосрочном планировании, то есть о диспетчеризации.

Стратегия планирования определяет, какие процессы мы планируем на выполнение для того, чтобы достичь поставленной цели. Известно большое количество различных стратегий выбора процесса, которому необходимо предоставить процессор. Среди них, прежде всего, можно назвать следующие стратегии:

- По возможности заканчивать вычисления (вычислительные процессы) в том же самом порядке, в котором они были начаты;
- Отдавать предпочтение более коротким процессам;
- Предоставлять всем пользователям (процессам пользователей) одинаковые услуги, в том числе и одинаковое время ожидания.

Дисциплины диспетчеризации

Известно большое количество правил (дисциплин диспетчеризации), в соответствии с которыми формируется список (очередь) готовых к выполнению задач. Различают два больших класса дисциплин обслуживания – бесприоритетные и приоритетные. При бесприоритетном обслуживании выбор задачи производится в некотором заранее установленном порядке без учета их относительной важности и времени обслуживания. При реализации приоритетных дисциплин обслуживания отдельным задачам предоставляется преимущественное право попасть в состояние исполнения. Перечень дисциплин обслуживания и их классификация приведены на рис.1.

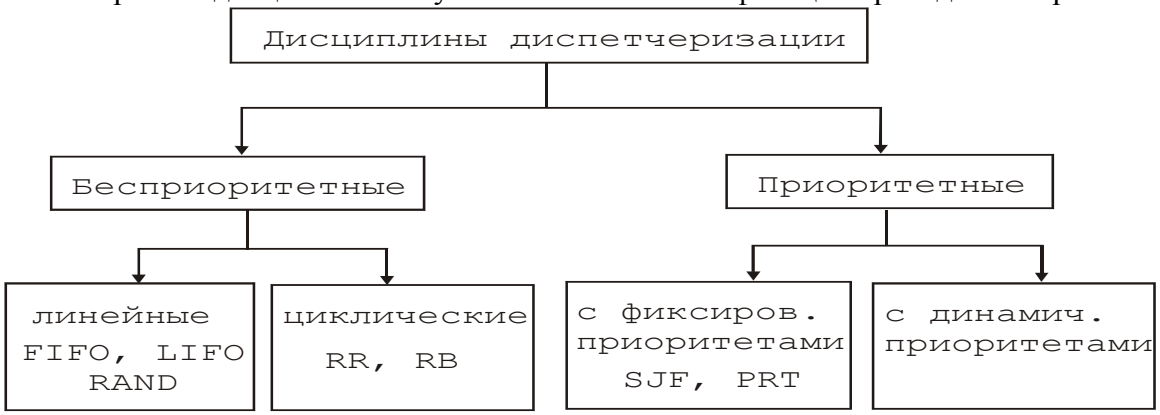


рис.1.

Виды дисциплин обслуживания:

1.FIFO - "первым пришел - первым выбран на обслуживание".
Время обслуживания заявки равно ее трудоемкости.
2.LIFO - "последним пришел - первым выбран на обслуживание".
Время обработки самой последней задачи аналогично FIFO.
3. RAND - случайный выбор заявки из очереди.
4. RR - "круговорот". Отличается от FIFO лишь временем обслуживания: каждая заявка получает определенный квант времени (одинаковый для всех).
5. PRT - выбор заявок на обслуживание по приоритету. Более приоритетной заявке соответствует меньшее число.
6.SJF - выбор заявки на обслуживание с минимальной трудоемкостью.

Нужно помнить о приоритетах следующее:

- Приоритет, присвоенный задаче, может являться величиной постоянной;
- Приоритет задачи может изменяться в процессе ее решения.

Качество диспетчеризации и гарантии обслуживания

Одна из проблем, которая возникает при выборе подходящей ДО, - это гарантия обслуживания. Дело в том, что при некоторых дисциплинах, например при использовании дисциплины абсолютных приоритетов, низкоприоритетные процессы оказываются обделенными многими ресурсами и, прежде всего, процессорным временем. Возникает реальная дискриминация низкоприоритетных задач, и ряд таких процессов, имеющих к тому же большие потребности в ресурсах, могут очень длительное время откладываться или, в конце концов, вообще могут быть не выполнены. Поэтому вопрос гарантии обслуживания является очень актуальным.

Более жестким требованием к системе, чем просто гарантированное завершение процесса, является его гарантированное завершение к указанному моменту времени или за указанный интервал времени. Существуют различные дисциплины диспетчеризации, учитывающие жесткие временные ограничения, но не существует дисциплин, которые могли бы предоставить больше процессорного времени, чем может быть в принципе выделено.

Планирование с учетом жестких временных ограничений легко реализовать, организуя очередь готовых к выполнению процессов в порядке возрастания их временных ограничений. Основным недостатком такого простого упорядочивания является то, что процесс (за счет других процессов) может быть обслужен быстрее, чем это ему реально необходимо. Для того чтобы избежать этого, проще всего процессорное время выделять все-таки квантами. Гарантировать обслуживание можно следующими тремя способами:

- Выделять минимальную долю процессорного времени некоторому классу процессов, если по крайней мере один из них готов к исполнению. Например можно отводить 20% от каждых 10мс процессам реального времени, 40% от каждых 2с – интерактивным процессам и 10% от каждых 5 мин – пакетным (фоновым) процессам;
- Выделять минимальную долю процессорного времени некоторому конкретному процессу, если он готов к выполнению;
- Выделять столько процессорного времени некоторому процессу, чтобы он мог выполнить свои вычисления к сроку.

Программная модель "Диспетчеризация задач".

Структурная схема, поясняющая принцип работы ЭВМ, представлена на рис.2.

Описание модели

Выделено три основных блока ЭВМ:
- генератор (GEN);
- супервизор (SV);
- центральный процессор (CPU).

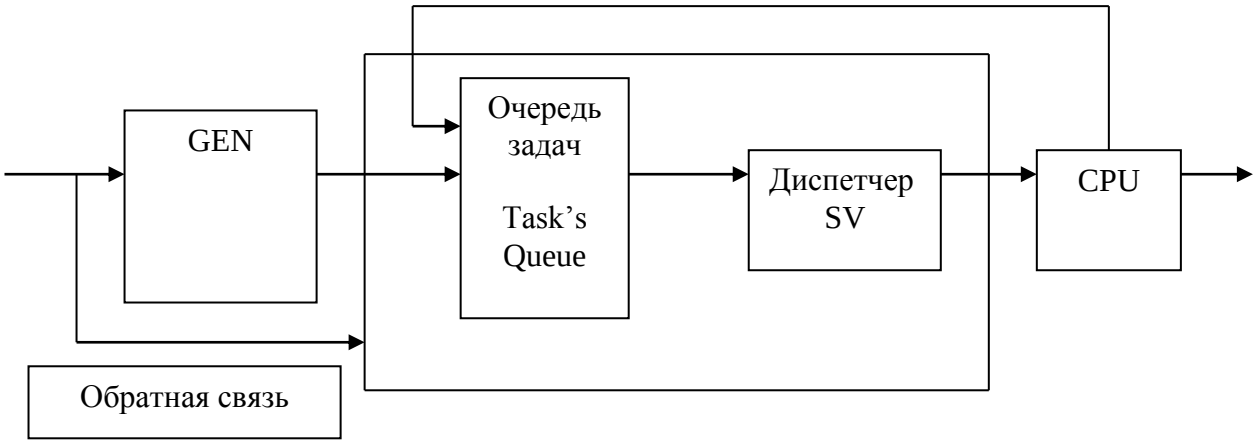


Рис.2. Структурная схема модели

Взаимодействие этих блоков и обеспечивает обработку задач (процессов).

Замечание: в модели ведется отсчет времени относительно модельного времени главных часов (TM). Наряду с TM вводятся понятия:

1. TSV - время работы SV;
2. TGEN - время работы GEN;
3. TCPU - время работы CPU над проблемной задачей (квант времени);
4. TTGEN - квант времени, по истечении которого блок генератора должен вступить в работу;
5. OST - остаток от кванта времени, отпущенного процессору для работы над задачей. (Время Доработки)

Блоком GEN формируются основные параметры задачи: номер задачи, приоритет, трудоемкость, время поступления в систему следующей задачи. Эти параметры формируются, исходя из максимально заданных значений входных величин генератора случайных чисел. Блок генератора вступает в работу по истечении TTGEN, при этом текущая работа прерывается и возобновляется после постановки вновь поступившей задачи в очередь на обработку.
Блок SV выполняет роль диспетчера задач.

Порядок работы модели.
1. Супервизор ожидает поступления в систему хотя-бы одной задачи.
2. После поступления задачи в систему ей выделяется квант процессорного времени. По истечении которого вновь будет запущен супервизор.
3. При очередном запуске супервизора происходит вызов планировщика заданий.
4. Пункты 2 и 3 повторяются до тех пор, пока задача не отработает отведенное ей время целиком.
5. Через некоторые промежутки времени запускается генератор новой задачи. При этом происходит запоминание текущего состояния системы. Генерация новой задачи. Вызов планировщика заданий. Возврат в запомненное состояние.
6. Весь процесс повторяется пока не будут выполнены все задачи.

Исходные данные

$X_i = [7 * X_{i-1} + 417] \bmod 1000;$

$K_i = [X_i / 7] \bmod 10$, $i=1 \div M$, $X_o = 011$ (последние три цифры зачетки),

M=10 – количество заданий

№	1	2	3	4	5	6	7	8	9	10
k_i	7	5	7	3	7	6	2	8	2	7
x_i	494	875	542	211	894	675	142	411	294	475
v_i	9	5	9	4	9	7	2	4	2	9
h_i	1	0	1	1	1	4	3	6	3	1
τ_i	30	30	30	10	30	20	40	40	40	30
t_i	0	7	12	19	22	29	35	37	45	47
$q*h_i$	5	1	5	5	5	20	15	30	15	5

где v_i – объем ОП, h_i – обьем внешней памяти, τ_i – трудоемкость, t_i – время поступления задания в систему, $q*h_i$ – время загрузки задания в систему (q=5).

Параметры системы:

V=16, H=12

Рабочие формулы:

$T_i = q*h_i + \tau_i$ - время обработки заданий, если бы они выполнялись по одному.

В случае обработки нескольких заданий, процессорное время распределяется поровну между заданиями.

Средневзвешенное время обращения:

$$W = \frac{\sum_{i=1}^{M} W_i}{M}, \text{ где } W_i = \frac{t_i^{\cdot} - t_i^{\cdot}}{T_i}$$

$\mathbf{t_i^з}$ - время завершения задания,

$\mathbf{t_i^п}$ - время поступления задания в систему.

Раздел №1

1. Временная диаграмма мультипрограммной работы ЭВМ

(дисциплина обслуживания FIFO (среди заданий в очереди, для которых достаточно свободных ресурсов, выбирается задание, поступившее первым))

Время	**События**	**V**	**H**
t=0	Не поступило ни одного задания на выполнение – простой системы.	16	12
t=7	Поступило задание 1, свободных ресурсов (ОП и ВУ) заданию 1 хватает, оно назначается на выполнение. Начинается ввод задания 1.	7	11
t=12	Поступило задание 2, свободных ресурсов заданию 2 хватает, оно назначается на выполнение. Начинается ввод задания 2. Окончен ввод задания 1. Проц-ное время отдается заданию 1.	2	11
t=13	Окончен ввод задания 2. Проц-ное время делится между заданиями 1 и 2.	2	11

t=19	Поступило задание 3, свободных ресурсов заданию 3 не хватает, оно становится в очередь.	2	11
t=22	Поступило задание 4, свободных ресурсов заданию 4 не хватает, оно становится в очередь.	2	11
t=29	Поступило задание 5, свободных ресурсов заданию 5 не хватает, оно становится в очередь.	2	11
t=35	Поступило задание 6, свободных ресурсов заданию 6 не хватает, оно становится в очередь.	2	11
t=37	Поступило задание 7, свободных ресурсов заданию 7 хватает, оно назначается на выполнение. Начинается ввод задания 7.	0	8
t=45	Поступило задание 8, свободных ресурсов заданию 8 не хватает, оно становится в очередь.	0	8
t=47	Поступило задание 9, свободных ресурсов заданию 9 не хватает, оно становится в очередь.	0	8
t=49	Поступило задание 10, свободных ресурсов заданию 10 не хватает, оно становится в очередь.	0	8
t=52	Окончен ввод задания 2. Проц-ное время делится между заданиями 1, 2 и 7.	0	8
t=81	Завершено выполнение задания 1. Ресурсы, занятые им, освобождены. Свободных ресурсов хватает чтобы назначить следующие задания. В работу вступает алгоритм FIFO и на выполнение назначается задание 3. Начинается ввод задания 3. Проц-ное время делится между заданиями 2 и 7.	0	8
t=83	Завершено выполнение задания 2. Ресурсы, занятые им, освобождены. Свободных ресурсов хватает чтобы назначить следующие задания. В работу вступает алгоритм FIFO и на выполнение назначается задание 4. Начинается ввод задания 4. Продолжается ввод задания 3. Проц-ное время отдается заданию 7.	1	7
t=86	Окончен ввод задания 3. Продолжается ввод задания 4. Проц-ное время делится между заданиями 3 и 7.	1	7
t=88	Окончен ввод задания 4.Проц-ное время делится между заданиями 3, 4 и 7.	1	7
t=118	Завершено выполнение задания 4. Ресурсы, занятые им, освобождены. Свободных ресурсов хватает чтобы назначить следующие задания. В работу вступает алгоритм FIFO и на выполнение назначается задание 8. Начинается ввод задания 8. Проц-ное время делится между заданиями 3 и 7.	1	2
t=148	Окончен ввод задания 8. Проц-ное время делится между заданиями 3, 7 и 8.	1	2
t=149	Завершено выполнение задания 7. Ресурсы, занятые им, освобождены. Так как свободных ресурсов заданию 9 хватает, оно назначается на выполнение. Начинается ввод задания 9 Проц-ное время делится между заданиями 3 и 8.	1	2
t=157	Завершено выполнение задания 3. Ресурсы, занятые им, освобождены. Свободных ресурсов хватает чтобы назначить следующие задания. В работу вступает алгоритм FIFO и на выполнение назначается задание 5. Начинается ввод задания 5. Продолжается ввод задания 9. Проц-ное время отдается заданию 8.	1	2
t=162	Окончен ввод задания 5. Продолжается ввод задания 9. Проц-ное время делится между заданиями 5 и 8.	1	2
t=164	Окончен ввод задания 9. Проц-ное время делится между заданиями 5, 8 и 9.	1	2

t=251	Завершено выполнение задания 5. Ресурсы, занятые им, освобождены. Так как свободных ресурсов заданию 10 хватает, оно назначается на выполнение. Начинается ввод задания 10. Проц-ное время делится между заданиями 8 и 9.	1	2
t=252	Завершено выполнение задания 8. Ресурсы, занятые им, освобождены. Продолжается ввод задания 10. Проц-ное время отдается заданию 9.	5	8
t=256	Окончен ввод задания 10. Проц-ное время делится между заданиями 9 и 10.	5	8
t=268	Завершено выполнение задания 9. Ресурсы, занятые им, освобождены. Так как свободных ресурсов заданию 6 хватает, оно назначается на выполнение. Начинается ввод задания 6. Проц-ное время отдается заданию 10.	0	7
t=288	Окончен ввод задания 6. Проц-ное время делится между заданиями 6 и 10.	0	7
t=296	Завершено выполнение задания 10. Ресурсы, занятые им, освобождены. Проц-ное время отдается заданию 6.	9	8
t=312	Завершено выполнение задания 6. Ресурсы, занятые им, освобождены. Работа системы завершена.	16	12

2. Временная диаграмма для алгоритма FIFO (см. рис 3)

Таблица, характеризующая выполнение заданий по FIFO.

№	1	2	3	4	5	6	7	8	9	10
Wi	2,11	2,37	3,94	6,40	6,34	6,93	2,04	2,96	4,02	7,06
tп	7	12	19	22	29	35	37	45	47	49
Нач. ввода	7	12	81	83	157	268	37	118	149	251
Нач. счета	12	13	86	88	162	288	52	148	164	256
Конец счета	81	83	157	118	251	312	149	252	268	296
Время на проц.	69	70	71	30	89	24	97	104	104	40

Максимальный коэффициент мультипрограммирования, который равен 3, был получен на участках: 52-81 (задания 1, 2 и 7); 88-118 (задания 3, 4 и 7); 148-149 (задания 3, 7 и 8); 164-251 (задания 5, 8 и 9).

Средневзвешенное время обращения W= 4,42

3. Временная диаграмма мультипрограммной работы ЭВМ

(дисциплина обслуживания SJF(среди заданий в очереди, для которых достаточно свободных ресурсов, выбирается задание с наименьшим τ_i))

Время	**События**	**V**	**H**
t=0	Не поступило ни одного задания на выполнение – простой системы.	16	12
t=7	Поступило задание 1, свободных ресурсов (ОП и ВУ) заданию 1	7	11

	хватает, оно назначается на выполнение. Начинается ввод задания 1.		
t=12	Поступило задание 2, свободных ресурсов заданию 2 хватает, оно назначается на выполнение. Начинается ввод задания 2. Окончен ввод задания 1. Проц-ное время отдается заданию 1.	2	11
t=13	Окончен ввод задания 2. Проц-ное время делится между заданиями 1 и 2.	2	11
t=19	Поступило задание 3, свободных ресурсов заданию 3 не хватает, оно становится в очередь.	2	11
t=22	Поступило задание 4, свободных ресурсов заданию 4 не хватает, оно становится в очередь.	2	11
t=29	Поступило задание 5, свободных ресурсов заданию 5 не хватает, оно становится в очередь.	2	11
t=35	Поступило задание 6, свободных ресурсов заданию 6 не хватает, оно становится в очередь.	2	11
t=37	Поступило задание 7, свободных ресурсов заданию 7 хватает, оно назначается на выполнение. Начинается ввод задания 7.	0	8
t=45	Поступило задание 8, свободных ресурсов заданию 8 не хватает, оно становится в очередь.	0	8
t=47	Поступило задание 9, свободных ресурсов заданию 9 не хватает, оно становится в очередь.	0	8
t=49	Поступило задание 10, свободных ресурсов заданию 10 не хватает, оно становится в очередь.	0	8
t=52	Окончен ввод задания 2. Проц-ное время делится между заданиями 1, 2 и 7.	0	8
t=81	Завершено выполнение задания 1. Ресурсы, занятые им, освобождены. Свободных ресурсов хватает чтобы назначить следующие задания. В работу вступает алгоритм SJF и на выполнение назначаются задания 4 и 9. Начинается ввод заданий 4 и 9. Проц-ное время делится между заданиями 2 и 7.	3	5
t=83	Завершено выполнение задания 2. Ресурсы, занятые им, освобождены. Свободных ресурсов хватает чтобы назначить следующие задания. В работу вступает алгоритм SJF и на выполнение назначается задание 6. Начинается ввод задания 6. Продолжается ввод заданий 4 и 9. Проц-ное время отдается заданию 7.	1	1
t=86	Окончен ввод задания 4. Продолжается ввод заданий 6 и 9. Проц-ное время делится между заданиями 4 и 7.	1	1
t=96	Окончен ввод задания 9. Продолжается ввод задания 6. Проц-ное время делится между заданиями 4, 7 и 9.	1	1
t=103	Окончен ввод задания 6. Проц-ное время делится между заданиями 4, 6, 7 и 9	1	1
t=114	Завершено выполнение задания 4. Ресурсы, занятые им, освобождены. Свободных ресурсов не хватает, чтобы назначить следующие задания. Проц-ное время делится между заданиями 6, 7 и 9.	5	2
t=163	Завершено выполнение задания 7. Ресурсы, занятые им, освобождены. Свободных ресурсов не хватает, чтобы назначить следующие задания. Проц-ное время делится между заданиями 6 и 9.	7	5
t=165	Завершено выполнение задания 6. Ресурсы, занятые им, освобождены. Свободных ресурсов хватает чтобы назначить	1	2

	следующие задания. В работу вступает алгоритм SJF и на выполнение назначаются задания 3 и 8. Начинается ввод заданий 3 и 8. Проц-ное время отдается заданию 9.		
t=170	Окончен ввод задания 3. Продолжается ввод задания 8. Проц-ное время делится между заданиями 3 и 9.	1	2
t=195	Окончен ввод задания 8. Проц-ное время делится между заданиями 3, 8 и 9.	1	2
t=196	Завершено выполнение задания 9. Ресурсы, занятые им, освобождены. Свободных ресурсов не хватает, чтобы назначить следующие задания. Проц-ное время делится между заданиями 3 и 8.	3	5
t=231	Завершено выполнение задания 3. Ресурсы, занятые им, освобождены. Свободных ресурсов хватает чтобы назначить следующие задания. В работу вступает алгоритм SJF и на выполнение назначается задание 5. Начинается ввод задания 5. Проц-ное время отдается заданию 8.	12	6
t=236	Окончен ввод задания 5. Проц-ное время делится между заданиями 5 и 8.	3	5
t=271	Завершено выполнение задания 8. Ресурсы, занятые им, освобождены. Свободных ресурсов не хватает, чтобы назначить следующие задания. Проц-ное время отдается заданию 5.	7	11
t=284	Завершено выполнение задания 5. Ресурсы, занятые им, освобождены. Так как свободных ресурсов заданию 10 хватает, оно назначается на выполнение. Начинается ввод задания 10. Нет заданий на обработку, образуется простой центрального процессора.	7	11
t=289	Окончен ввод задания 10. Проц-ное время отдается заданию 10	7	11
t=319	Завершено выполнение задания 10. Ресурсы, занятые им, освобождены. Работа системы завершена.	16	12

4. Временная диаграмма для алгоритма SJF (см рис 4)

Таблица, характеризующая выполнение заданий по SJF.

№	1	2	3	4	5	6	7	8	9	10
Wi	2,11	2,37	6,06	6,13	7,29	3,25	2,29	3,23	2,71	7,71
tп	7	12	19	22	29	35	37	45	47	49
Нач. ввода	7	12	165	81	231	83	37	165	81	284
Нач. счета	12	13	170	86	236	103	52	195	96	289
Конец счета	81	83	231	114	284	165	163	271	196	319
Время на проц.	69	70	61	28	48	62	111	76	100	30

Максимальный коэффициент мультипрограммирования, который равен 4, был получен на участке: 103-114 (задания 4, 6, 7 и 9).
Средневзвешенное время обращения W= 4,32

5. Выводы

1. Максимальный коэффициент мультипрограммирования у FIFO равен 3, у SJF равен 4.
2. Дисциплина обслуживания SJF обладает меньшим значением средневзвешенного времени обращения (W=4,32) по сравнению с FIFO (W=4,42), следовательно, для такой последовательности заданий выгоднее использовать дисциплину обслуживания SJF. Это объясняется тем, что в системе заданий с малой трудоемкостью больше чем продолжительных.
3. При использовании дисциплины обслуживания SJF на участке 284-289 происходит простой центрального процессора.

Раздел №2

1. Задание

Разработать структуру функционирования диспетчера работ в вычислительной системе, заданной в разделе 1, на интервале T1, который соответствует максимальному коэффициенту мультипрограммирования. Квант времени, выделяемый каждой работе, выбирается исходя из конкретной ситуации: число работ, параллельно занимающих процессор, интервал T1, дисциплины обслуживания.

Диспетчер использует метод разделения времени в сочетании с приоритетами.

2. Исходные условия

Вариант № 5

Дисциплины обслуживания:

- бесприоритетные ДО (БП): LIFO;
- приоритетные ДО (П): обслуживание с динамическим приоритетом (зависимость от времени ожидания - $t_{ож}$).

Участок, на котором коэффициент максимален, я беру из временной диаграммы Раздел №1. Это участок 103-114, его длительность 11 единиц времени.

Приоритет, зависимый от времени ожидания, задается в файле spo.dat:

104 --- // Выбранный участок

4 --- // Коэффициент мультипрограммирования – количество задач, параллельно выполняемых на процессоре.

20 --- приоритет задачи 4, t ожид. = 59 единиц;

30 --- приоритет задачи 6, t ожид. = 48 единиц;

90 --- приоритет задачи 7, t ожид. = 0 единиц;

40 --- приоритет задачи 9, t ожид. = 34 единиц.

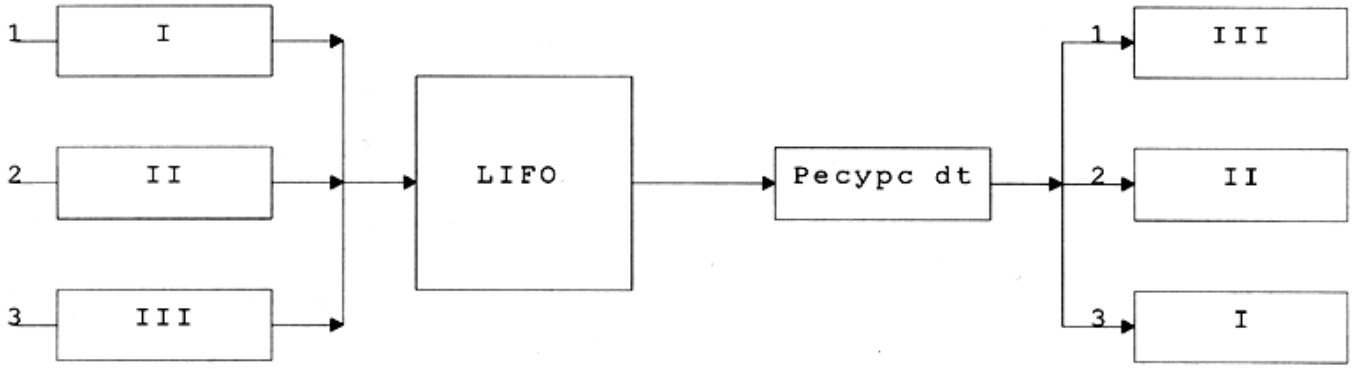
3. Диспетчеризация задач для бесприоритетной ДО LIFO

Общий принцип функционирования бесприоритетных ДО таков: дисциплина обслуживания выбирает заявки из очереди без учета их важности, по принципу последовательньости их поступления.

В данном случае используется ДО LIFO (LAST INPUT – FIRST OUTPUT), т.е последним пришел – первым обслужился. На этом принципе и основывается построение оптимизированной таблицы задач в данном случае. Так как у системы нет информации о приоритетах, то t обсл. почти постоянно.

Программно это реализуется следующим образом: в цикле проверяется входной номер задачи и делается перестановка соответственно алгоритму LIFO. Число циклов соответствует числу задач. Таким образом, просматриваются все входящие задачи и составляется оптимизированная таблица. Этот этап называется *верхним уровнем управления процессами.*

4. Схема диспетчера бесприоритетной ДО LIFO



Описание схемы:

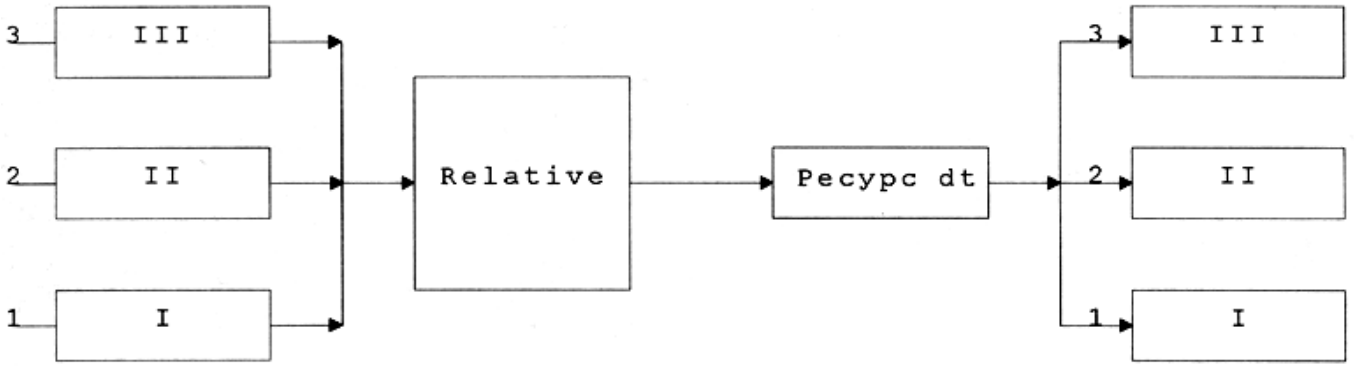
1. На входе 3 задания (I, II, III) в порядке поступления (1, 2, 3).
2. В соответствии с ДО LIFO формируется оптимизированная таблица задач.
3. Задаче выделяется процессорное время.
4. Выходная цепочка задачи построена в соответствии с ДО.

Количество заданий не обязательно 3, может быть от 2-х и более.

5. Диспетчеризация задач для приоритетной ДО с динамическим приоритетом

При поступлении более приоритетной задачи в систему прерывания не происходит, заявка на своем уровне выполняется до конца. Если имеется две или более заявок одинакового приоритета, то они выполняются по принципу FIFO. При формировании очереди задач рекомендуется пропустить вперед задания с малым временем ожидания. И оптимизация исходной таблицы задач производилась именно по этому принципу.

6. Схема диспетчера приоритетной ДО с динамическим приоритетом



Описание схемы:

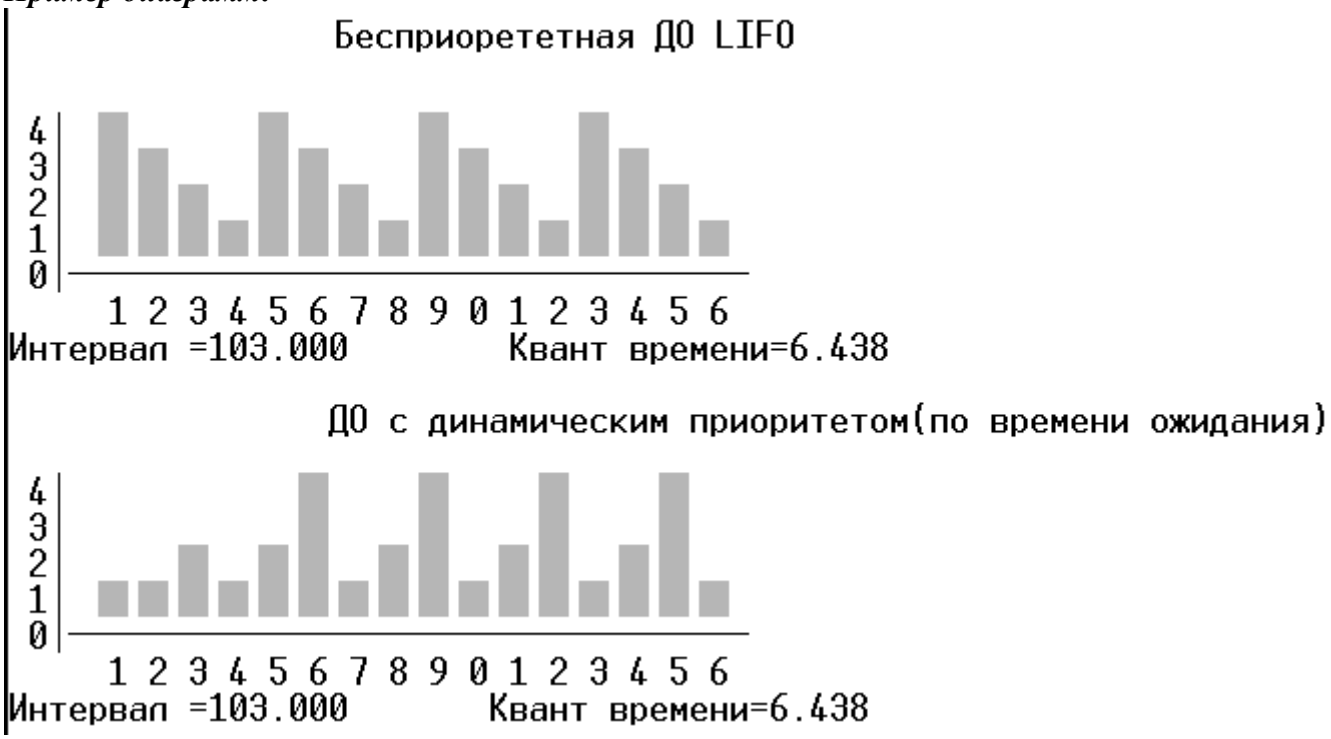
1. На входе 3 задания (I, II, III) в порядке поступления (1, 2, 3).
2. В соответствии с ДО с относительным приоритетом при поступлении более приоритетной задачи в систему прерывания не происходит, заявка на своем уровне выполняется до конца, а после включается более приоритетная. Если имеется несколько заявок одинакового приоритета, то они выполняются по принципу FIFO.
3. Задаче выделяется процессорное время.
4. Выходная цепочка задачи построена в соответствии с ДО.

Количество заданий не обязательно 3, может быть от 2-х и более.

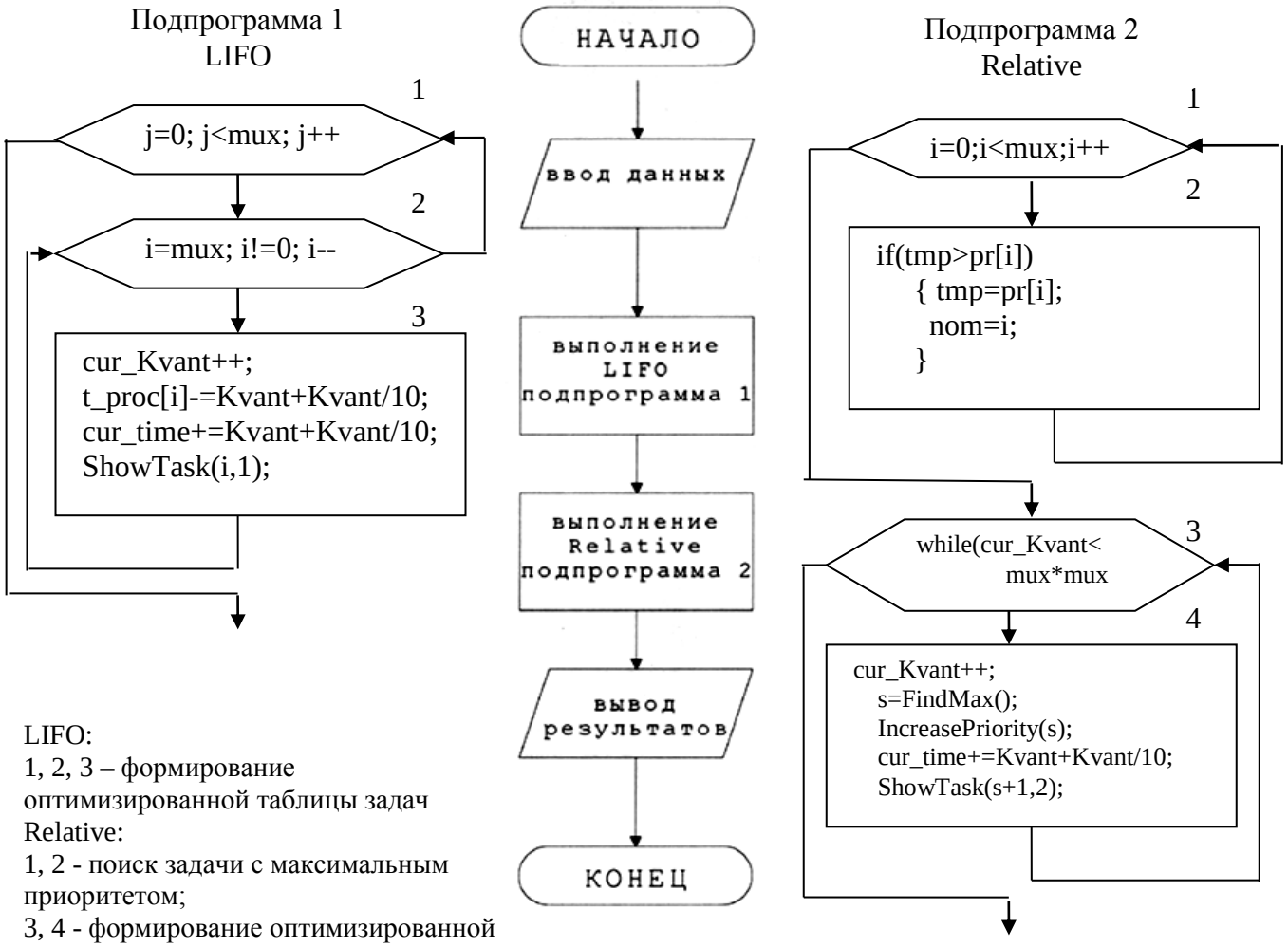
7. Окончание работы диспетчеров

По окончании работы каждого из диспетчеров мы видим на экране столбчатые диаграммы, характеризующие распределение задач в оптимизированной таблице.

Пример диаграмм:



8. Алгоритм функционирования диспетчера



LIFO:
1, 2, 3 – формирование оптимизированной таблицы задач
Relative:
1, 2 - поиск задачи с максимальным приоритетом;
3, 4 - формирование оптимизированной таблицы задач

9. Текст программы диспетчера

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#include <stdlib.h>

#define OS    9
#define TEXT  0
#define FRAPH 11
#define ERROR 12

void DrawCoordinates(int);
void ShowTask(int,int);
int  FindMax(void);
void IncreasePriority(int);
void DO_LIFO(void);
void DO_DINAMIC(void);

FILE *in_file;     // файл с задачами
FILE *text;        // файл для результатов
float t;           // время
int mux;           //
int *t_proc;
float Kvant;       // квант времени на задачу
int cur_Kvant=0;   // текущий квант
float cur_time=0;  // текущее время
int i,j;
int *pr;
int s;
//---------------------------------------------------------------------------
//Функция рисования координат
void DrawCoordinates(int y)
{ textcolor(OS);
  // Рисование оси x
  for(i=1;i<mux*mux*2+5;i++)
   { gotoxy(i+1,y*10+1);
     cprintf("─");
   }
 // Подпись у оси x
 textcolor(TEXT);
 for(i=1;i<mux*mux+1;i++)
 { gotoxy(i*2+4,y*10+2);
   cprintf("%d",i%10);
 }
 // Рисование оси y и подпись у оси
 textcolor(OS);
 for(i=0;i<mux+1;i++)
 { gotoxy(2,y*10-i+1);
   cprintf("%d│",i);
 }
 gotoxy(1,1);
}
//---------------------------------------------------------------------------
// Печать гистограмы текущего задания
void ShowTask(int nom,int y)
{ textcolor(FRAPH);
  for(i=0;i<nom;i++)
  { gotoxy(cur_Kvant*2+3,y*10-i);
    cprintf(" ■");
  }
 fprintf(text,"Передать управление SV t=%.3f   \t",cur_time-Kvant/10);
 fprintf(text,"Поступила задача %d t=%.3f\n",nom,cur_time);
```

```
}
//---------------------------------------------------------------------
//Поиск задачи с максимальным приоритетом
int FindMax(void)
{ int tmp=pr[0];
  int nom=0;
  for(i=0;i<mux;i++)
   { if(tmp>pr[i])
      { tmp=pr[i];
        nom=i;
      }
   }
  return nom;
}
//---------------------------------------------------------------------
// Повышение приоритета при ДО с динамическим приоритетом
void IncreasePriority(int nom)
{ for(i=0;i<mux;i++)
   { if(i==nom)
       continue;
     pr[i]--;
   }
}
//---------------------------------------------------------------------
//Дисциплина обслуживания LIFO
void DO_LIFO(void)
{ textcolor(TEXT);
  fprintf(text,"\t\tБесприорететная ДО LIFO\n");
  cprintf("               Бесприорететная ДО LIFO\n");
  for(j=0;j<mux;j++)
   { for(i=mux;i!=0;i--)
      { cur_Kvant++;
        t_proc[i]-=Kvant+Kvant/10;
        cur_time+=Kvant+Kvant/10;
        ShowTask(i,1);
      }
   }
 fprintf(text,"\nИнтервал =%.3f\tКвант времени=%.3f",t,Kvant);
 textcolor(TEXT);
 cprintf("\r\n\n\nИнтервал =%.3f        Квант времени=%.3f",t,Kvant);
}
//---------------------------------------------------------------------
// Дисциплина обслуживания с динамическим приоритетом (по времени ожедания)
void DO_DINAMIC(void)
{ cur_Kvant=0;
  cur_time=0;
  textcolor(TEXT);
  fprintf(text,"\n\n\t\tДО с динамическим приоритетом(по времени ожидания)\n");
  cprintf("\r\n\n               ДО с динамическим приоритетом(по времени ожидания)\n");
  while(cur_Kvant<mux*mux)
   { cur_Kvant++;
     s=FindMax();
     IncreasePriority(s);
     cur_time+=Kvant+Kvant/10;
     ShowTask(s+1,2);
   }
 fprintf(text,"\nИнтервал =%.3f\tКвант времени=%.3f",t,Kvant);
 textcolor(TEXT);
 cprintf("\r\n\n\nИнтервал =%.3f       Квант времени=%.3f",t,Kvant);
}
//---------------------------------------------------------------------
void main(int argc,char *argv[])
{ textbackground(7);
```

```
  clrscr();
  textcolor(ERROR);
  if(argc==1)
  { if((in_file=fopen("spo.dat","r"))==NULL)
     { cprintf("Ошибка открытия файла spo.dat");
       exit(1);
     }
  }
  else
   if((in_file=fopen(argv[1],"r"))==NULL)
    { cprintf("Ошибка открытия файла %s",argv[1]);
      exit(1);
    }
  if((text=fopen("text","w+"))==NULL)
   cprintf("Ошибка открытия файла результатов text");
  fscanf(in_file,"%f",&t);
  fscanf(in_file,"%d",&mux);
  if((t_proc=new int[mux+1])==NULL)
   cprintf("Не хватает памяти");
  if((pr=new int[mux+1])==NULL)
   cprintf("Не хватает памяти");
  for(i=0;i<mux;i++)
   { fscanf(in_file,"%d",&t_proc[i]);
     pr[i]=t_proc[i]/10;
   }
  Kvant=t/(mux*mux);
  DrawCoordinates(1);
  DrawCoordinates(2);
  DO_LIFO();
  DO_DINAMIC();
}
```

Заключение

По окончании работы каждого из диспетчеров мы видим на экране столбчатые диаграммы, характеризующие распределение задач в оптимизированной таблице. Общий принцип функционирования бесприоритетных ДО таков: дисциплина обслуживания выбирает заявки из очереди без учета их важности, по принципу последовательньости их поступления.

Список литературы

1. Лихачева Г.Н., Медведева В.Д. Операционные системы. - М.:Статистика, 1980, - 23с.
2. Основы теории вычислительных систем / Под ред. С.А.Майорова. - М.:Высшая школа, 1978, - 408с.
3. Бек А. Введение в системное программирование: Перев. с англ. - М.:Мир, 1988, - 448с.
4. Планирование верхнего уровня управления заданиями. Методическое указание к лабораторной работе по курсу СПО (ОВП). N336.13 – НЭТИ, 1990 г.