Контрольная работа 1-01

Вариант 1
(решение)

0. За разговоры с соседом -3 балла за каждый.

1. (6 баллов) Возможны ли следующие переходы процесса из одного состояния в другое?

a. Из состояния *готовность* в состояние *исполнение*.
b. Из состояния *ожидание* в состояние *исполнение*
c. Из состояния *готовность* в состояние *ожидание*

Если переход возможен, кратко сформулируйте, когда он происходит. Если невозможен, напишите почему.

Решение:

a. Переход возможен и происходит, когда планировщик процессов принимает решение выбрать данный процесс для исполнения из очереди готовых процессов.
b. Переход невозможен, так как в состояние *исполнение* процесс может попасть только из состояния *готовность*. В состоянии *ожидание* процесс ждет наступления какого-либо события в системе, без которого он не может продолжать свою работу, т.е. исполняться.
c. Переход не возможен, так как в состояние ожидание может попасть только исполняющийся процесс.

Оценка:

По одному баллу за каждый правильный ответ о возможности или невозможности перехода и по 1 баллу за каждое грамотное обоснование.

2. (3 балла) Кратко сформулируйте разницу между программой и процессом.

Решение:

Программа представляет собой статический объект - исполняемый файл. Процесс - это динамический объект, совокупность последовательно исполняемых инструкций и связанных с ними ресурсов системы (стеки, основное адресное пространство, выделенные устройства ввода-вывода и т.д.), над которым операционная система может совершать определенные операции. Таким образом, понятие процесса не включается в понятие программы. С другой стороны понятие программа также не включается в понятие процесса, так как для решения задачи одна программа может породить несколько процессов.

3. (12 баллов) Пусть в вычислительную систему поступают пять процессов различной длительности по следующей схеме:

Номер процесса	Время поступления в систему	Время исполнения
1	3	4
2	7	8
3	4	6
4	10	1
5	0	5

Вычислите среднее время между стартом процесса и его завершением (*turnaroud time*) и среднее время ожидания процесса (*waiting time*) для каждого из трех алгоритмов планирования FCFS (First Come First Served), RR (Round Robin) и SJF (Short Job First). При вычислениях считать, что процессы не совершают операций ввода-вывода, величину кванта времени принять равной 1, временем переключения контекста пренебречь. Процесс, поступающий в систему, считать готовым к исполнению в момент поступления. Для алгоритма RR принять, что вновь прибывший процесс помещается в начало очереди процессов, готовых к исполнению, и, следовательно, сразу выбирается на исполнение.

Решение:

a. Рассмотрим выполнение процессов в системе для алгоритма FCFS. По вертикали в таблице отложены номера процессов, по горизонтали - моменты времени. Буква И означает состояние исполнения, буква Г - состояние готовности.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1				Г	Г	И	И	И	И															
2								Г	Г	Г	Г	Г	Г	Г	Г	И	И	И	И	И	И	И	И	
3					Г	Г	Г	Г	Г	И	И	И	И	И	И									
4											Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	И
5	И	И	И	И	И																			

Среднее время между стартом процесса и его завершением: tt = (6 + 16 + 11 + 14 + 5)/5 = 10.4.
Среднее время ожидания: wt = (2 + 8 + 5 + 13 + 0)/5 = 5.6.

b. Рассмотрим выполнение процессов в системе для алгоритма RR:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1				И	Г	Г	И	Г	Г	Г	Г	И	Г	Г	И									
2								И	Г	Г	Г	Г	И	Г	Г	И	Г	И	Г	И	Г	И	И	И
3					И	Г	Г	Г	И	Г	Г	Г	Г	И	Г	Г	И	Г	И	Г	И			
4											И													
5	И	И	И	Г	Г	И	Г	Г	Г	И														

Среднее время между стартом процесса и его завершением: tt = (12 + 17 + 17 + 1 + 10)/5 = 11.4.
Среднее время ожидания: wt = (8 + 9 + 11 + 0 + 5)/5 = 6.6.

c. Рассмотрим выполнение процессов в системе для алгоритма SJF:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1				Г	Г	И	И	И	И															
2								Г	Г	Г	Г	Г	Г	Г	Г	Г	И	И	И	И	И	И	И	И
3					Г	Г	Г	Г	Г	И	И	И	И	И	И									
4											Г	Г	Г	Г	Г	И								
5	И	И	И	И	И																			

Среднее время между стартом процесса и его завершением: tt = (6 + 17 + 11 + 6 + 5)/5 = 9.
Среднее время ожидания: wt = (2 + 9 + 5 + 5 + 0)/5 = 4.2.
Возможны решения для вытесняющего алгоритма с временами (6+17+12+1+5)/5=8.2 и (2+9+6+0+0)/5=3.4 соответственно.

Оценка:

По 2 балла за каждое правильно вычисленное время.

4. (3 балла) Дайте краткое определение термина "критическая секция процесса".

Решение:

Критическая секция процесса - это участок процесса, в котором процесс использует ресурсы, разделяемые с другими процессами, таким образом, что одновременное использование ресурсов разными процессами может привести к недетерминированным (изменяющимся от запуска к запуску) результатам.

5. (18 баллов) Рассмотрим следующее программное решение для прохождения процессами своих критических участков:

***int c1 = 1, c2 = 1, turn;* /* Разделяемые переменные */**

Процесс *P1*:

```
while(1){

        turn = 1;
        c1 = 0;
        while (!c2 && turn == 1);
        < критический участок 1>
        c1 = 1;
        < другие операции 1>

}
```

Процесс *P2*:

```
while(1){

        turn = 2;
        c2 = 0;
        while (!c1 && turn == 2);
        < критический участок 2>
        c2 = 1;
        < другие операции 2>

}
```

Всем ли требованиям к алгоритмам синхронизации (mutual exclusion, progress, bound waiting) удовлетворяет данное решение? Обоснуйте свои утверждения.

Решение:

a. Начнем с bounded waiting - отсутствия бесконечного ожидания для входа процесса в свой критический участок. Допустим, процесс *P1* ждет входа в критический участок, т.е. находится внутри цикла *while (!c2 && turn == 1);* Это означает, что *turn == 1* и *c2 == 0*. Так как *turn == 1*, то процесс *P2* не будет ожидать входа в критический участок. После не более чем одного прохождения критического участка процесс *P2* выполнит операцию *turn = 2* и процесс *P1* сможет войти в свой критический участок. Условие bounded waiting (ограниченного ожидания) выполняется.

b. Progress - те процессы, которые находятся вне критических участков (а также вне их пролога и эпилога) не должны препятствовать другим процессам входить в их критические участки. Если процесс *P2* выполняет *<другие операции 2>,* то значение переменной *c2 == 1* и никто не препятствует процессу *P1* войти в критический участок. Условие progress выполняется.
c. Взаимоисключение. Покажем, что это условие не выполняется:

```
P1: turn = 1;
P2: turn = 2;
P2: c2 = 0;
P2: while(!c1 && turn == 2); // c1 != 0 !!!!
P2: <вход в критический участок>
P1: c1 = 0;
P1: while(!c2 && turn == 1); // turn == 2 !!!!
P1:<вход в критический участок>
```

Оценка:

По 6 баллов за каждое условие.

6. (5 баллов) В вычислительной системе со страничной организацией памяти из 32-х бит адреса старшие 6 его бит отводятся для номера страницы.

a. Какое максимальное количество страниц может иметь процесс? Каков размер страницы?
b. Для некоторого процесса таблица страниц имеет вид:

Номер страницы	Начальный адрес
0	0x4000000
1	0xC000000

2	0x8000000
3	0x0000000

Каким физическим адресам соответствуют адреса виртуальной памяти 0x1234, 0x10000000, 0x03ab00de?

Решение:

1. На номер страницы отводится 6 бит, значит максимальное количество страниц у процесса 2**6 = 64 страницы. На смещение внутри страницы остается 32 - 6 = 26 бит, поэтому размер страницы 2**26 байт.
2. 0x1234 -> 0x4001234, 0x10000000 -> error, 0x03ab00de -> 0x7ab00de.

Оценка:

По 1 баллу за каждое верное значение

7. (9 баллов) Для некоторого процесса известна следующая строка запросов страниц памяти

7, 2, 0, 3, 2, 0, 3, 1, 2, 7, 4, 1, 7, 0, 7

Сколько ситуаций отказа страницы (*page fault*) возникнет для данного процесса при каждом из трех алгоритмов замещения страниц - FIFO (Fist Input Fist Output) , LRU (the Least Recently Used), OPT (optimal) , если процессу выделено 3 кадра памяти?

Решение:

a. Начнем с FIFO:

Строка запросов	7	2	0	3	2	0	3	1	2	7	4	1	7	0	7
Страницы в памяти	7	2	0	3	3	3	3	1	2	7	4	1	1	0	7
		7	2	0	0	0	0	3	1	2	7	4	4	4	0
			7	2	2	2	2	0	3	1	2	7	7	1	4
Page fault	+	+	+	+				+	+	+	+	+		+	+

Количество page fault'ов - 11.

b. Для LRU:

Строка запросов	7	2	0	3	2	0	3	1	2	7	4	1	7	0	7
Страницы в памяти	7	2	0	0	0	0	0	0	2	2	2	1	1	1	1
		7	2	2	2	2	2	1	1	1	4	4	4	0	0
			7	3	3	3	3	3	3	7	7	7	7	7	7
Page fault	+	+	+	+				+	+	+	+	+		+	

Количество page fault'ов - 10.

c. Для OPT:

Строка запросов	7	2	0	3	2	0	3	1	2	7	4	1	7	0	7
Страницы в памяти	7	2	0	0	0	0	0	0	0	0	4	4	4	0	0
		7	2	2	2	2	2	2	2	7	7	7	7	7	7
			7	3	3	3	3	1	1	1	1	1	1	1	1
Page fault	+	+	+	+				+		+	+			+	

Количество page fault'ов - 8.

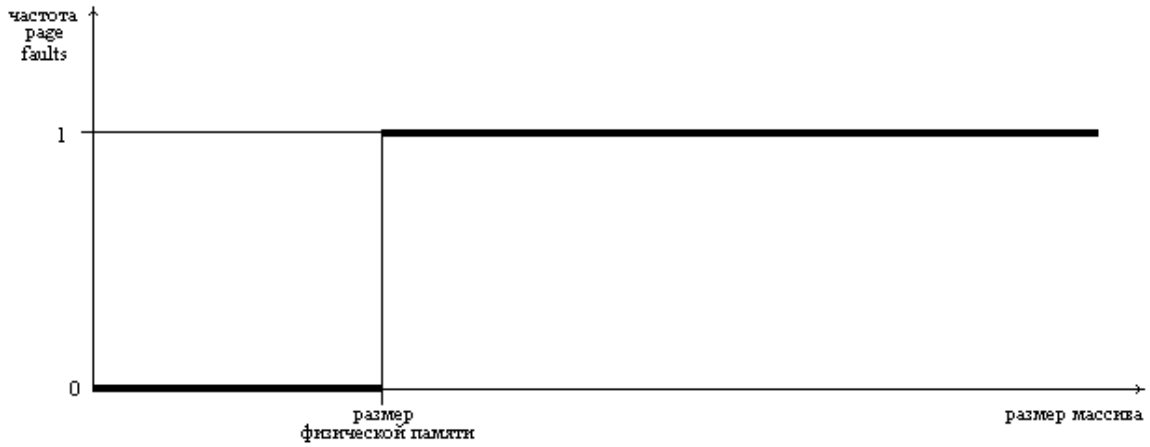
Оценка:

По 3 балла за каждое значение

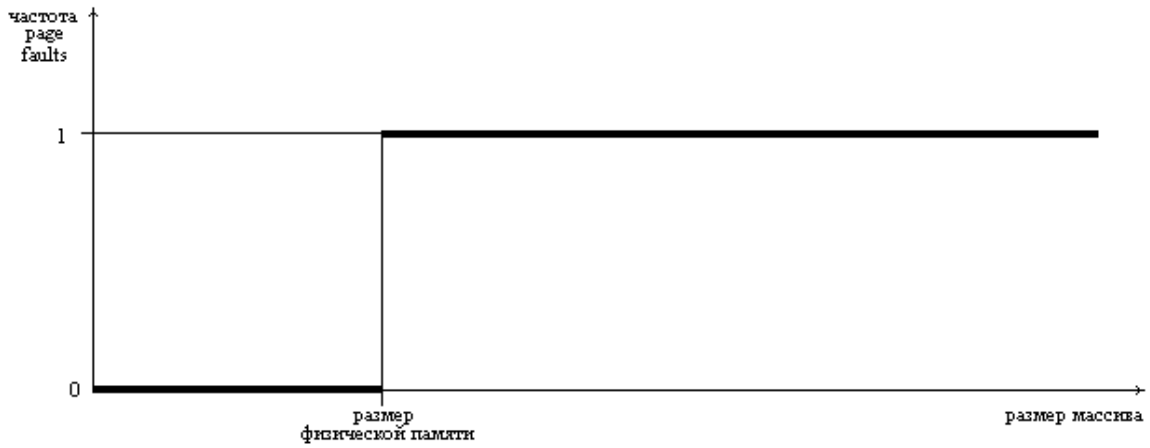
8. **(15 баллов) Рассмотрим вычислительную систему со страничной организацией памяти, на которой выполняется одна программа, несколько раз последовательно сканирующая большой одномерный массив данных (это означает, что для массива, занимающего, например, в памяти четыре страницы, строка запроса страниц выглядит следующим образом: 1, 2, 3, 4, 1, 2, 3, 4, ...). Для каждого из трех алгоритмов замещения страниц - FIFO (First Input First Output), LRU (the Least Recently Used) и OPT (OPTimal) нарисуйте график зависимости частоты page faults от размеров массива. По оси y откладывается отношение числа page faults к длине строки запроса (если массив, занимающий четыре страницы памяти, сканируется 2 раза, то длина строки запроса равна 8), по оси x - размер массива: от размеров, требующих одну страницу, и до размеров, значительно превышающих размер физической памяти. Опишите наиболее характерные точки на графике.**

Решение:

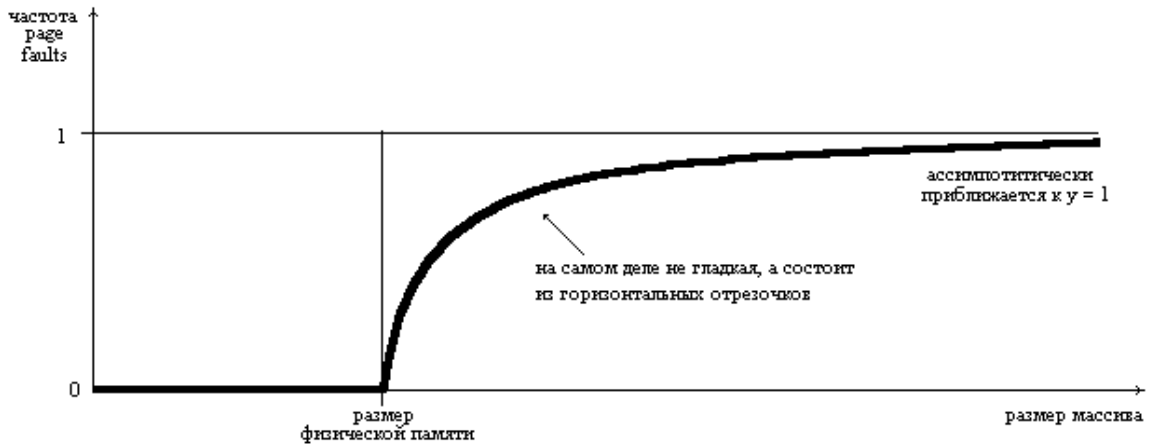
a. Для FIFO:



9. Для LRU:



10. Для OPT:



Оценка:

По 5 баллов за каждый правильный график. Не указано, что точка отхода от 0 значения частоты соответствуют совпадению размеров массива с размером физической памяти - -1балл для каждого графика. Не указано, что кривая для алгоритма OPT ассимптотически стремится к прямой y = 1 при x, стремящемся к бесконечности, - -1 балл