Как делать оси

ещё есть репозиторий задачек из лекций

Экзамен длится ровно одну пару, то есть 80 минут.
Тут используется такая же система с процентами что была в лабах.

Баллы	Оценка
6	4
8	5
10	6
12	7
14	8
16	9
18	10

1. Теоретический вопрос третьего модуля (2 балла)

пока я предлагаю молится
примерный список вопросов есть в пуде (надо сначало войти в lms)
вот табличка ответов на них. её сделал Кирилл💚
также есть книжка карпова, в которой есть примерно всё and then some

2. Теоретический вопрос четвёртого модуля (2 балла)

или можно распечатать весь конспект всех лекций рукописным шрифтом
есть ещё такой вариант конспекта

3. Задача про page fault (2 балла)

Page fault – это когда процесс пытается достать страницу памяти, но она ещё не подгружена в физическую память. Не стоит путать с сегфолтом. Сегфолт происходит когда процесс пытается достать память, которая не существует вообще. А page fault – не ошибка: процесс просто ждёт пока система подгрузит ему эту страницу, и ни в коем случае не падает.

Пример

Вот пример этой задачки из пуда:

> Для некоторого процесса известна следующая строка запросов страниц памяти 1, 5, 2, 3, 2, 1, 4, 5, 1, 2, 3, 4, 1, 5, 2, 1, 3, 2, 4.
> Сколько ситуаций отказа страницы (page fault) возникнет для данного процесса при использовании алгоритма замещения страниц FIFO (First Input First Output), если процессу выделено 3 кадра памяти?

Решение:

Память			Запрос	Page fault
1			1	☑
1	5		5	☑
1	5	2	2	☑
3	5	2	3	☑
3	5	2	2	☐
3	1	2	1	☑
3	1	4	4	☑
5	1	4	5	☑
5	1	4	1	☐
5	2	4	2	☑
5	2	3	3	☑
4	2	3	4	☑
4	1	3	1	☑
4	1	5	5	☑
2	1	5	2	☑
2	1	5	1	☐
2	3	5	3	☑
2	3	5	2	☐
2	3	4	4	☑

Ответ: 15
(в этой табличке изначально заполнена колонка “запрос” из условия)

Алгоритмы

1. FIFO (First In First Out) – для каждой страницы создается метка времени и в очередь
2. OPT (optimal) – для каждой странице вычисляется количество команд до первого использования и наибольшая уходит. (Невозможный, тк смотрит в будущее)
3. LRU (Least Recently Used) – убирается тот который дольше всего не использовался. (Трудоемкий)
4. NFU (Not Frequently Used) – убирается самый редко использующийся. Требует доп счетчики (кажется на лекциях не было - Саша)

4. Задача про tlb (2 балла)

тут нужно позвать Алису и спросить у неё что это вообще за задача такая¿
upd: Алиса тоже ничего не понимает
про tlb есть такая формула в конспектах
Ассоциативная память TLB
t (среднее) = t1 + h*t0 + (1-h)(C+1)t0
где t0 – время доступа к оперативке
t1 – время доступа к tlb
h – hit ratio (вероятность того, что страница есть в TLB)

C – количество уровней организации (например, если пишут про двухуровневую страничную схему, то C = 2 - Саша)
Ассоциативная память (translation lookaside buffer TLB) – быстрый кэш который обеспечивает хранение части таблицы. Это эффективно так как зачастую 1 процесс использует небольшую часть таблицы

Пример

> Известно, что время доступа к оперативной памяти 100 нс, а время доступа к ассоциативной памяти – 10 нс. Частота попаданий в ассоциативную память при обращении к данным (hit ratio) составляет 90%. Чему равно среднее время обращения за данным к памяти для одноуровневой таблицы страниц?

t = 10 + 0.9·100 + (1-0.9)·(1+1)·100 = 120 нс

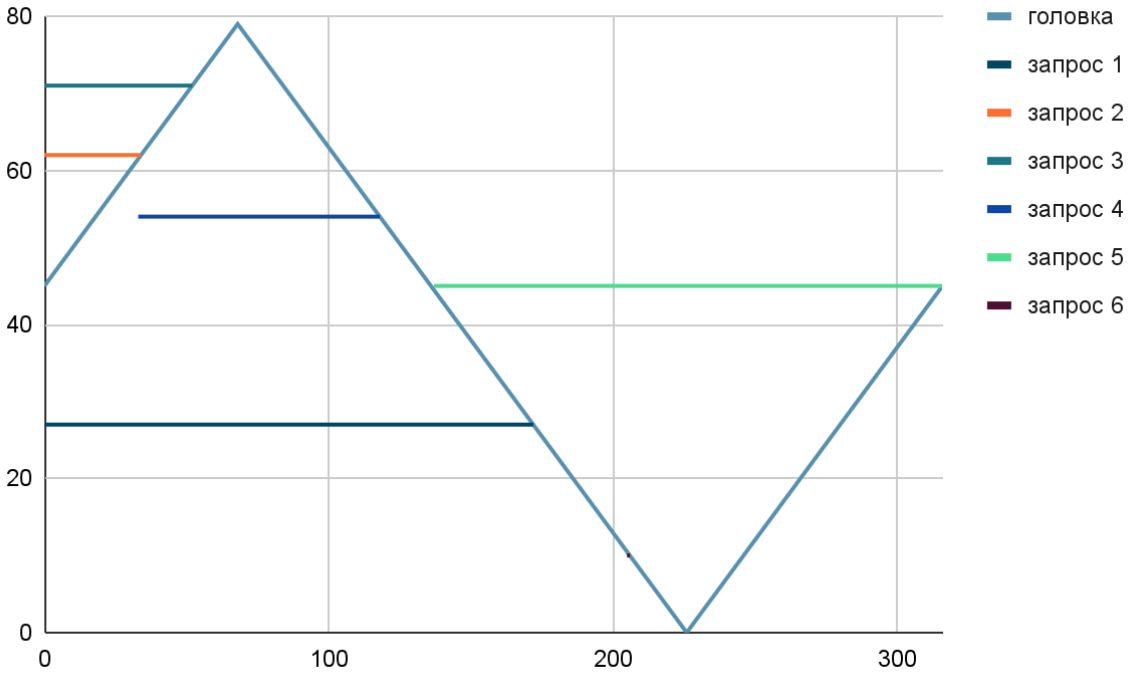
5. Задача про диск (3 балла)

Эта задача была разобрана на консе. (это ссылка на правильное время)

Пример

> Пусть у нас есть диск, насчитывающий 80 цилиндров.
> Время перемещения головок между соседними цилиндрами составляет 2 мс.
> В начале – на 45-м цилиндре.
> Алгоритм SCAN, начальное направление – в сторону увеличения номеров.

номер	27	62	71	54	45	10
время	0	0	0	33	137	205

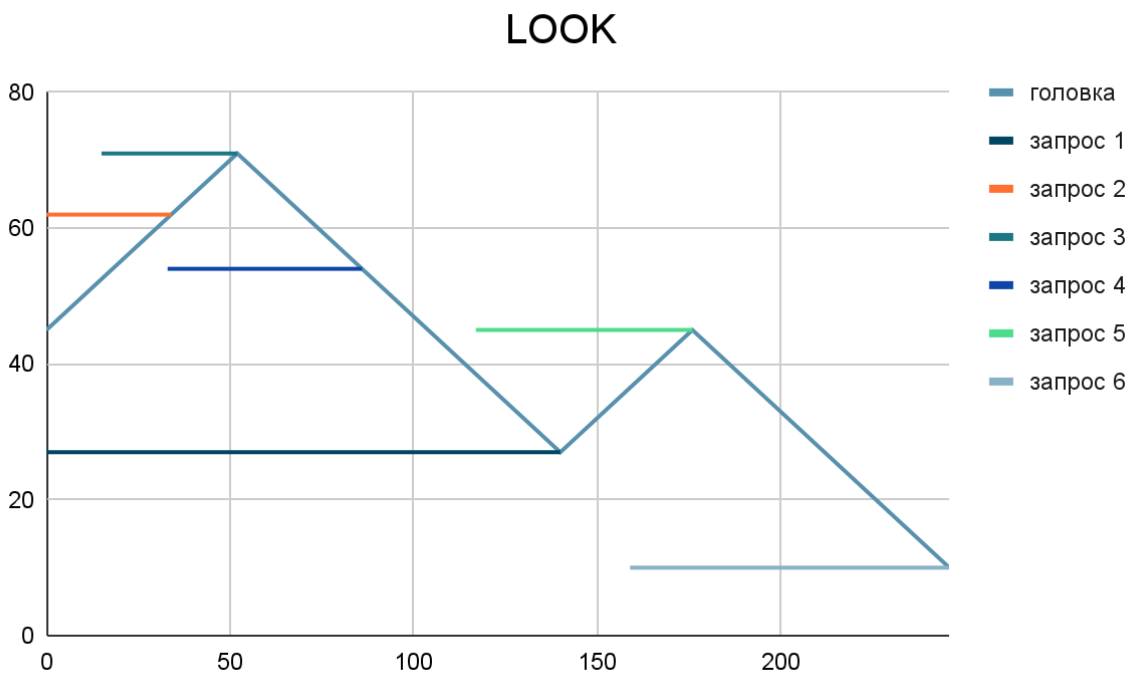


В этом примере очень много off by one ошибок: запрос 6 появляется за 1 мс до его обработки, а запрос 5 опаздывает на 1 мс.
надо ещё расписать арифметику для всех этих пересечений

Алгоритмы

1. FCFS (First come first serve) – в очереди FIFO
2. SSTF (Short Seek Time First) – обслуживание ближайшего, при равенстве расстояний можно по разному, например FCFS
3. SCAN – из стороны в сторону до обоих концов
4. LOOK – из стороны в сторону, но не доходя до конца если не требуется
5. C SCAN – циклический скан, при достижении конца быстрое перемещение в другой конец
6. C Look – циклический look при достижении последнего запроса быстрое перемещение в другой конец ленты к первому запросу (100% не будет, Карпов сказал на консе + на лекциях не было)

я тут хотел сделать красивое описание с графиками для каждого алгоритма



Сначала мы обработали запрос 2, и потом смотрим есть ли запросы выше него.
Идём обрабатывать запрос 3 и выше него ничего нет, поэтому мы разворачиваемся и идём вниз обрабатывать запрос 4.
Ниже запроса 4 есть запрос 1.
А вот после запроса 1 запрос 6 ещё не появился, поэтому мы сначала обработаем запрос 5.

6. Задача про сегфолты (3 балла)

Эта задача была разобрана на консе.
В этой задаче мы переводим виртуальные адреса в физические.

У нас виртуальная память разбита на сегменты а каждый сегмент разбит на страницы.
Кадр – это физический адрес страницы.

Пример

> В вычислительной системе с сегментно-страничной организацией памяти и 32-х битовой адресацией максимальный размер сегмента составляет 8 MB, а размер страницы памяти 256 KB (1 MB = 1024 KB, 1 KB = 1024 байта). Для некоторого процесса в этой системе таблица сегментов, находящихся в памяти, имеет вид:

Номер сегмента	Длина сегмента
0	`0x1f0000`
1	`0x230000`

Таблицы страниц, находящихся в памяти, для сегментов 0 и 1 приведены ниже:

Сегмент 0		Сегмент 1	
Номер стр.	Кадр (дес.)	Номер стр.	Кадр (дес.)
0	12	1	32
6	22	2	63
9	15	7	22

Каким физическим адресам соответствуют логические адреса: `0x009f2810`, `0x001c1511`, `0x00262214`?

У нас 32 битный адрес нужно разбить на номер сегмента, номер страницы и адрес внутри страницы.

8MB = 2^{23} байт и поэтому для адреса внутри сегмента нужно 23 бита, и для номера остаётся 32-23=9 бит.

256KB = 2^{18} байт и поэтому для адреса внутри страницы нужно 18 бит, и для номера остаётся 23-18=5 бит.

`0b 00000000 00000000 00000000 00000000`
Тут есть номер сегмента, номер страницы и адрес внутри страницы.
PS: позиции битов надо считать с конца. красные это не первые 9 бит, а все кроме последних 23

Для `0x009f2810` номер сегмента 1, а адрес внутри сегмента `0x1f2810`.
`0x1f2810` < `0x230000` поэтому в сегмент мы попадаем.
Номер страницы получается 7 а смещение в ней `0x32810`.
Такая страница у нас есть и “кадр” у неё 22=`0x16`.
Теперь, чтобы найти физический адрес, нам нужно склеить `0x16` и `0x32810`.
`0b 00000000 01011011 0x2810`
Ответ: `0x005b2810`

Для `0x001c1511` номер сегмента 0, а адрес внутри сегмента `0x1c1511`.
`0x1c1511` < `0x1f0000` поэтому в сегмент мы попадаем.
Номер страницы получается 7, а такой страницы нет.
Ура у нас ✨сегфолт✨. (то есть page fault)

Для `0x00262214` номер сегмента 0, а адрес внутри сегмента `0x262214`.
`0x262214` ≥ `0x1f0000` в сегмент мы НЕ попадаем.
Ура у нас ✨сегфолт✨. (только наверно на экзе лучше писать не “сегфолт” а “ошибка сегментации”)

PS: для тех кто за два года так и не выучил си, `0x` это шестнадцатеричная система, а `0b` двоичная (в джаве так же; и даже в сишарпе © Егор)

7. Задача про радужную таблицу (6 баллов)

Эта задача была разобрана на консе.
Алгоритм краткосрочного планирования отвечает за то какой из процессов, для которых уже выделена память, будет сейчас исполняться.
Алгоритм долгосрочного планирования решает в каком порядке раздавать память процессам. Если у него не получилось поместить какой-то процесс, то он просто сдаётся и не пытается поместить процесс более низкого приоритета, даже если для него есть место (см пример 1 процесс 5).
Стратегия размещения определяет куда именно нужно запихнуть память уже конкретного выбранного процесса.

Пример 1

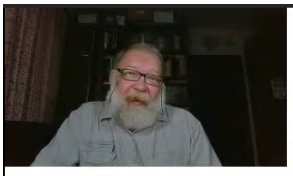
> Рассмотрим однопроцессорную одноядерную вычислительную систему с объемом свободной оперативной памяти 220 MB, в которой используется схема организации памяти с динамическими (переменными) разделами. Для долгосрочного планирования процессов в ней применен алгоритм планирования с приоритетами.
>
> В систему поступают пять заданий с различной длительностью, различным объемом занимаемой памяти и различными приоритетами по следующей схеме:

Номер задания	Момент поступления в очередь	Время исполнения	Объем занимаемой памяти (MB)	Приоритет
1	0	6	60	2
2	0	3	100	1
3	3	4	60	3
4	4	3	80	1
5	5	2	20	2

> Нарисуйте диаграмму использования оперативной памяти, вычислите среднее время между стартом задания и его завершением (turnaround time) и среднее время ожидания (waiting time) для следующей комбинации алгоритма краткосрочного планирования и стратегии размещения процессов в памяти:
> вытесняющий SJF (Shortest Job First) и worst fit (наименее подходящий);
>
> При вычислениях считать, что процессы не совершают операций ввода-вывода. Временами переключения контекста, рождения процессов и работы алгоритмов планирования

пренебречь. Освобождение памяти, занятой процессами, происходит немедленно по истечении их CPU burst. Краткосрочное планирование осуществляется после рождения новых процессов в текущий момент времени. Наивысшим приоритетом в системе является приоритет со значением 0.

Буквы:
Г – готовность
И – исполнение
О – ожидание
wt = sum(О, Г) / количество процессов
tt = sum(О, Г, И) / количество процессов



Планирование процессов и память

t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
P_1	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	И	И	И	И	И	И
P_2	И	И	И															
P_3				И	И	И	И											
P_4					О	О	О	Г	Г	И	И	И						
P_5						О	О	И	И									

$$tt = \frac{18+3+4+(3+5)+(2+2)}{5} = 7.4$$

100	100	100	60	60	60	60	80	80	80	80	80	100	100	100	100	100	100	100
			40	40	40	40	20	20	20	20	20							
60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60
60	60	60	60	60	60	60	20 40	20 40	60	60	60	60	60	60	60	60	60	60

Пример 2

Рассмотрим однопроцессорную вычислительную систему с объемом оперативной памяти 200 Mb, в которой используется схема организации памяти с динамическими (переменными) разделами. Для долгосрочного планирования процессов в ней применен алгоритм SJF. В систему поступают пять заданий с различной длительностью и различным объемом занимаемой памяти по следующей схеме:

Номер задания	Момент поступления в очередь	Время исполнения (CPU burst)	Объем занимаемой памяти (MB)
1	0	3	80
2	2	4	50
3	3	5	60
4	4	2	80
5	5	1	10

Нарисуйте диаграмму расположения процессов в оперативной памяти для различных моментов времени, вычислите среднее время между стартом задания и его завершением

(turnaround time) и среднее время ожидания (waiting time) для следующей комбинации алгоритма краткосрочного планирования и стратегии размещения процессов в памяти: RR (Round Robin) и worst fit (наименее подходящий);

При вычислениях считать, что процессы не совершают операций ввода-вывода, величину кванта времени принять равной ~~4~~ 1. Временами переключения контекста, рождения процессов и работы алгоритмов планирования пренебречь. Освобождение памяти, занятой процессами, происходит немедленно по истечении их CPU burst. Краткосрочное планирование осуществляется после рождения новых процессов в текущий момент времени. Для алгоритма RR принять, что родившиеся процессы добавляются в САМЫЙ конец очереди готовых процессов (ПОСЛЕ процесса, перешедшего в состояние готовность из состояния исполнение в это время).

У нас в радужной табличке 3+4+5+2+1=15 колонок:

t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P_1	И	И	И												
P_2			Г	И	Г	И	Г	Г	И	Г	И				
P_3				Г	И	Г	И	Г	Г	И	Г	И	Г	И	
P_4					О	О	О	О	О	О	О	Г	И	Г	И
P_5						Г	Г	И							

200	80	80	80	60	60	60	60	60	60	60	60	60	60	60	60
				20	20	20	20	20	20	20	20	80	80	80	80
	120	120	50	50	50	50	50	50	50	50	50				
			70	70	70	10	10	10	70	70	70	60	60	60	60
						60	60	60							

tt = (3+9+11+11+2)/5 = 7.2
wt = (5+6+9+1)/5 = 4.2

Пример 2.1

я в предыдущей задаче неправильно прочитал условие и не заметил что квант времени 4 а не 1

t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P_1	И	И	И												
P_2			Г	И	И	И	И								
P_3				Г	Г	Г	Г	И	И	И	И	Г	И		
P_4					О	О	О	О	О	О	О	О	Г	И	И

P_5						Г	Г	Г	Г	Г	Г	И			

200	80	80	80	60	60	60	60	60	60	60	60	60	60	60	60
				20	20	20	20	70	70	70	70	70	80	80	80
	120	120	50	50	50	50	50								
			70	70	70	10	10	10	10	10	10	10			
						60	60	60	60	60	60	60	60	60	60

Алгоритмы планирования

1. First Come First Served (FCFS) – работают в том порядке в котором пришли
2. Robin Round (RR) – карусель процессов, закольцованный FCFS
3. Shortest Job First (SJF) – вытесняющий и не вытесняющий приоритет у процесса с меньшим CPU burst
4. Приоритетное планирование – вытесняющее и невытесняющее, по приоритету

Стратегии размещения в памяти

1. First fit – первый подходящий раздел
2. Best fit – в тот раздел который более как раз (в самый маленький)
3. Worst fit – в тот что менее всего как раз (в самый большой)

Авторы

© Костя Борисов
© Алиса Сальникова
© Саша Агроскин
© доц. Карпов Владимир Ефимович

Special thanks to

© The color Red
© The color Green
© The color Blue
© 0x from the C programming language
© The concept of tables
© The “✨” emoji