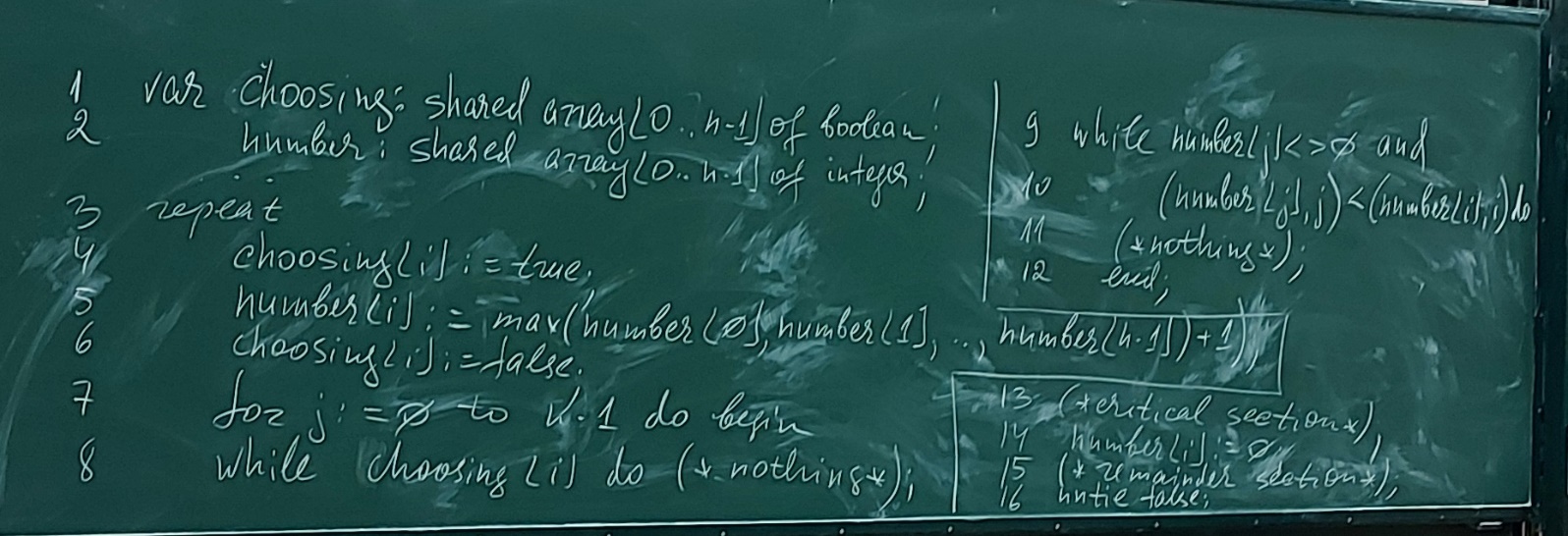
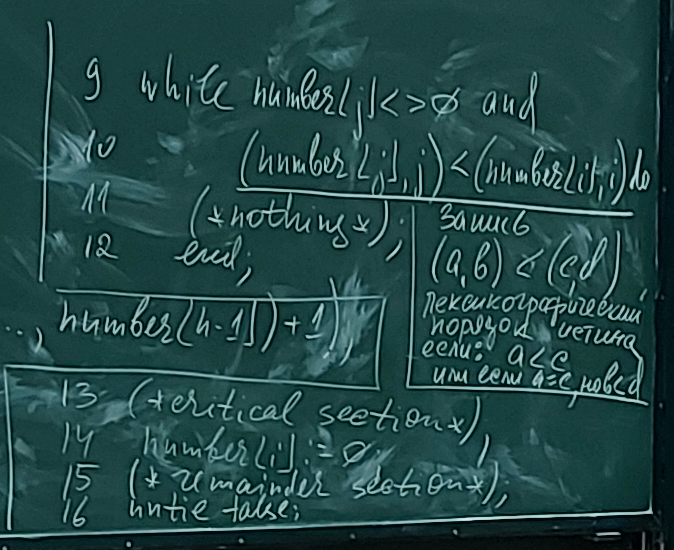
# Операционные системы Лекция 14? (29.11.2021)



Если процесс выбирает номер в то время когда процесс PI пытается также найти номер, то PI будет ждать до тех пор пока другие процессы не сделают это. (Строка 8)



Взаимоисключение, ухудшает общую производительносить системы т.к. блокировки приводят к тому что другие параллельыне операции не могут быть выполнены.

### Проблема спящего парикмахера

Это классическая задача синхронизации и меж процессорного взаимодействия в многозадачной операционной системе. Аналогия обоснована на гипотетической парикмахерской с одним парикмахером. У него одно кресло и приёмная со стульями. Когда он заканчивает обслуживать клиента, он его отпускает и идёт в приёмную посмотреть есть ли ждущие. Если есть то берёт одного, сажает его в кресло и начинает его обслуживать. Если никого нет, то он сам садится в кресло и спит. Каждый клиент, когда приходит смотрит на парикмахера. Если парикмахер спит, то будит его и садится в кресло. Если парикмахер занят, то пытается сесть на стул в приёмной. Если свободного стула нет, то уходит.

Все проблемы связаны с тем фактом, что действия парикмахера и клиента асинхронные. То есть время, которое тратится каждым клиентом и парикмахером на обслуживание и проверки – разное, непредсказуемое.

Например: клиент может зайти и увидеть, что парикмахер работает и пойти в приёмную. Парикмахер за это время может закончить и прийти в приёмную раньше клиента => оба ждут.

Другой пример: два клиента могут прийти одновременно, при этом в приёмной только 1 свободный стул. Пытаются его занять => конфликт.

Эта задача ставится и в процессе рассуждения мы начинаем глубже понимать проблему взаимодействия асинхронных процессов.

Мы рассмотрим отдельный/специальный вид взаимодействия процессов со специальным названием – рандеву.

Ада – язык параллельного программирования, разработан был для военных целей. (Для моделирования военных действий). Именно в языке Ада появился рандеву.

?Задачи связываются между собой при помощи входов?  
task <имя> is  
 entry <идентификатор входа>(дискретный диапазон)(формальная часть);  
 <другие объявления входов>

Если одна задача выдала сообщение ко входу другой задачи, то обе задачи теряют свою независимость т.к. они установили рандеву. И до тех пор, пока рандеву действует, задачи синхронизированы. В механизме рандеву отсутствует симметрия. Это значит, если одна задача запрашивает рандеву с другой задачей, то другая задача может принять это рандеву, а может отказаться. То есть рандеву происходит только тогда, когда задача ко входу которой было обращения принимает этот вызов. Поэтому в языке Ада есть оператор accept.

accept:

accept<идентификатор входа>(выражение) формальная часть  
 do последовательность операторов end;

Очень важно св. рандеву – одновременное получение … последовательности операторов (без блокировок). Именно поэтому этот способ выделяется как отдельны и имеет собственное название.

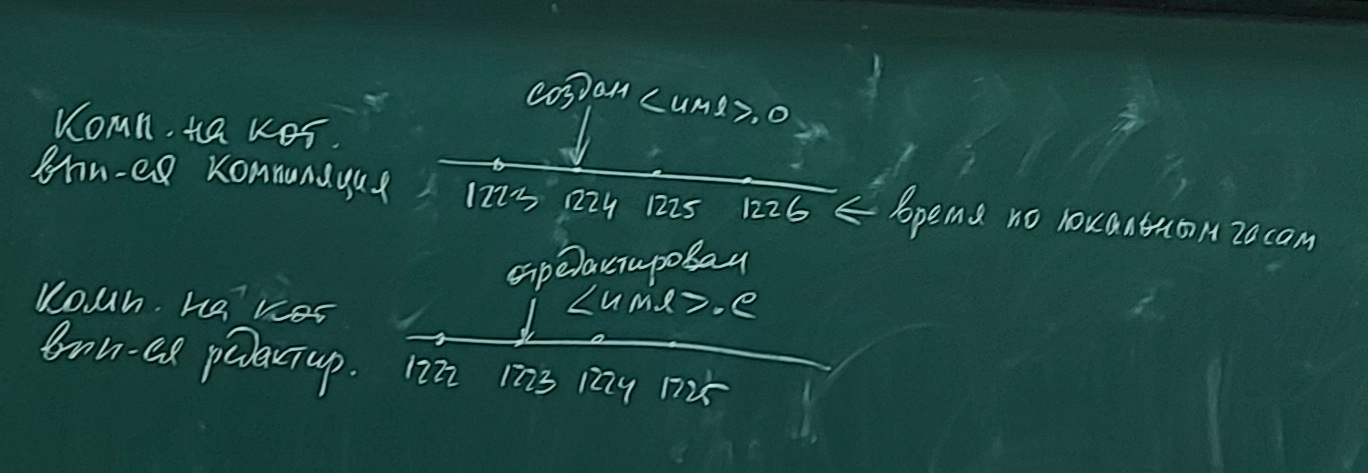
### Синхронизация часов

Рассмотрим ?программную реализацию make?

Обычно в UNIX большие программы разделаются на несколько файлов.

Программа make проверяет время последней модификации всех исходных и объектных файлов. Если исходный файл имеет время последней модификации больше времени соответствующего ему объектного файла, то программа make считает, что его нужно перекомпилировать.

На отдельно стоящей машине нет проблем. Скажем на одном компьютере выполняется редактирование, а на втором компиляция. Каждый компьютер имеет свой счётчик тиков.



… От 215 998 до 216 002 …

… средняя солнечная секунда mean solar second

1949 г. – изобретены атомные часы …

TAI – International Atomic Time

Солнечный день в настоящее время удлиняется. TAI остаётся не изменным => по TAI полдень будет наступать всё позже. Проблему решили потерянными секундами leap seconds. В результате мир перешёл на универсально согласованное время UTC – Universal Coordinated Time.

Чтобы не было расхождения по времени в Америке есть университет времени в штате Колорадо.

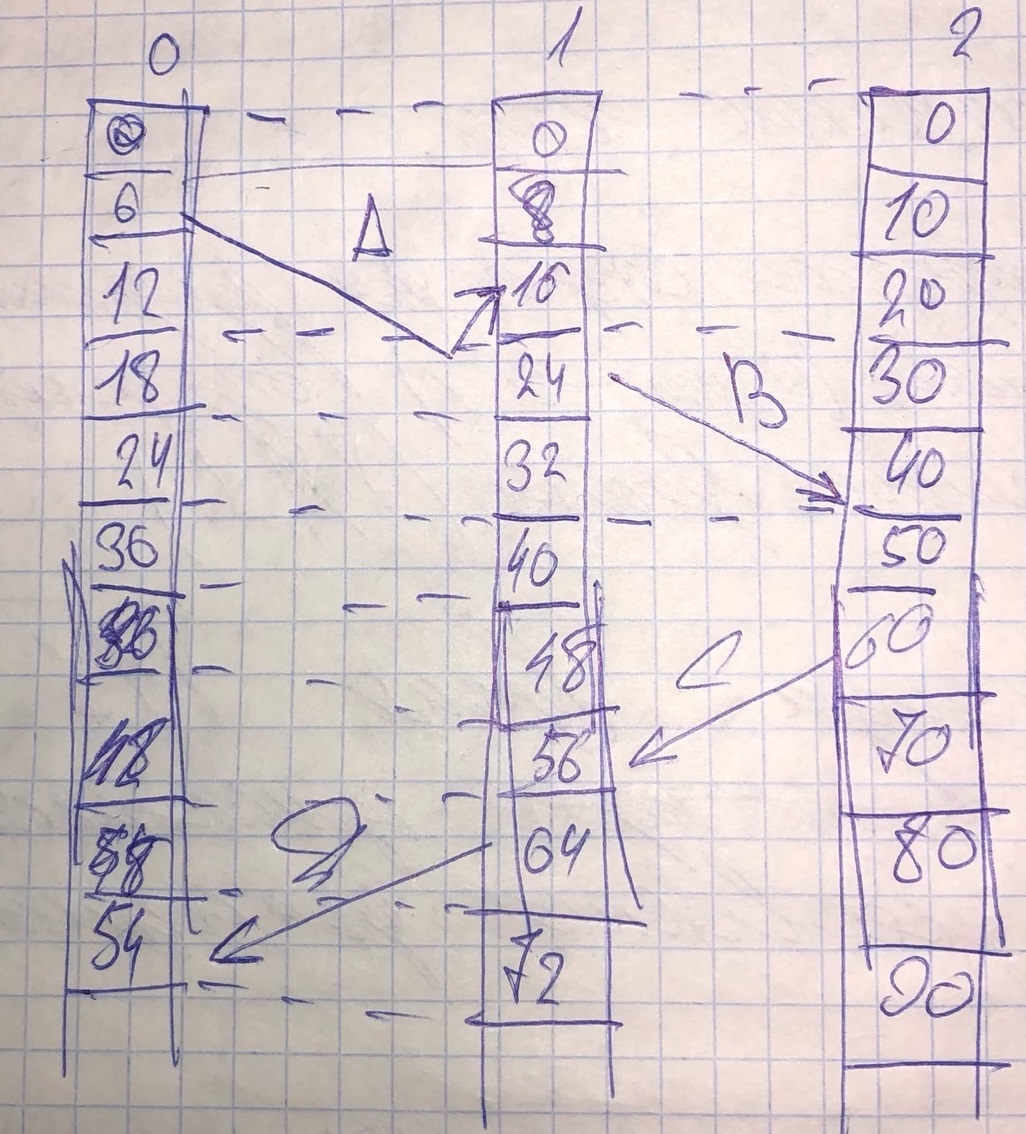
Существует целый ряд алгоритмов синхронизации по физическим часам.

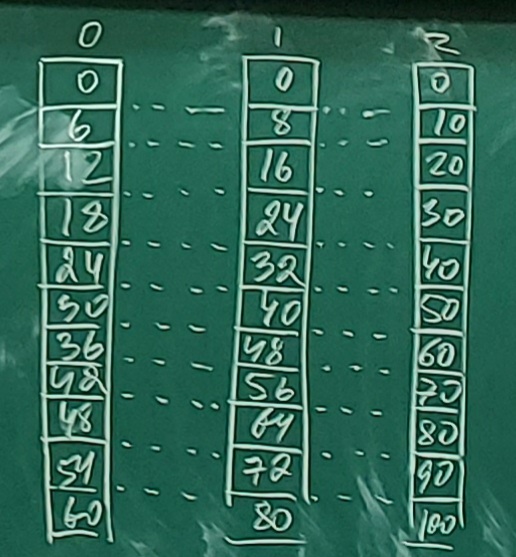
1. алгоритм Кристиана;
2. алгоритм Беркли;
3. усредняющий алгоритм.

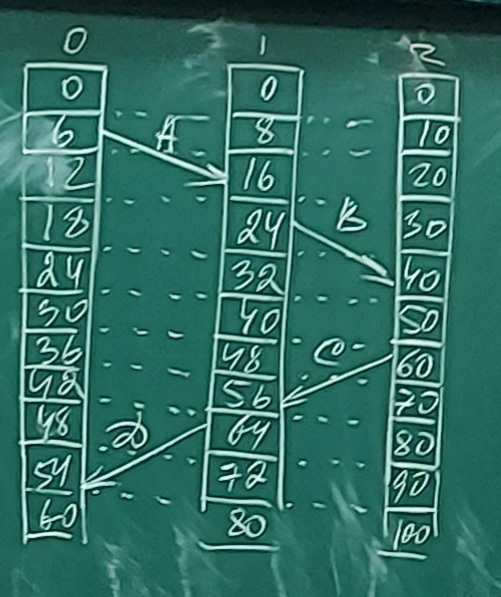
Важно соблюдать соотношение, которое определяется как “случилось до”/ “случилось после”.

В 1978 г. Лампортом был предложен алгоритм синхронизации логических часов.

Суть алгоритма, что каждому сообщению приписывается время отправки по локальным часам.

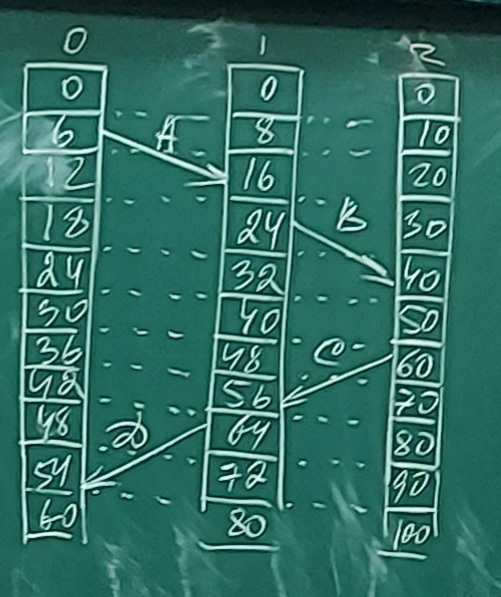






Алгоритм Лампорта применяется для синхронизации временных ответок (для соблюдения отношения "случилось до / случилось после")

* Процесс, отправивший сообщение, так же отправляет время отправки по локальным часам.
* В случае, если время процесса меньше, чем время отправления, то он сам устанавливает время отправления: пришедшее + 1.



Данный алгоритм обеспечивает правильную нумерацию событий, и является самым распространённым решением задачи временной синхронизации по логическим часам.

Так или иначе синхронизация по логическим часам широко используется. Существуют и другие алгоритмы.

## Алгоритмы взаимоисключения в … системах