БИЛЕТ № 23

1) Монолитные ОС. Недостатки и достоинства.

Монолитная ОС – набор процедур, каждая из которых может вызывать другие когда нужно. Каждая процедура имеет строго определенный интерфейс в терминах параметров и результатов, и каждая имеет возможность вызывать любую другую для выполнения некоторой необходимой для нее работы.

По способу построения ядра системы - монолитное ядро или микроядерный подход. ОС использующие монолитное ядро, компонуются как одна программа, работающая в привилегированном режиме и использующая быстрые переходы с одной процедуры на другую, не требующие переключения из привилегированного режима в пользовательский и наоборот. При построении ОС на базе микроядра, работающего в привилегированном режиме и выполняющего только минимум функций по управлению аппаратурой, функции более высокого уровня выполняют специализированные компоненты ОС – программные серверы, работающие в пользовательском режиме. При таком построении ОС работает более медленно, так как часто выполняются переходы между привилегированным режимом и пользовательским, но система получается более гибкой и ее функции можно модифицировать, добавляя или исключая серверы пользовательского режима;

2) Переход процесса в заблокированное состояние.

Доп. Инфа. Условие перехода процесса из активного состояния в заблокированное.

Процесс, которому выделяется время процессора, переводиться в активное состояние или состояние выполнения.

Во время выполнения процесса ему могут понадобиться дополнительные ресурсы, но для их получения необходимо некоторое время. Т.е. процесс должен ожидать наступления некоторого события или окончания выполнения конкретной операции (например, ввода/вывода для получения/выдачи данных). Такой процесс переводится в заблокированное состояние.

Доп. Инфа.
Условие перехода процесса из заблокированного состояния в готовое.

Если активный процесс не выполнился за выделенный ему квант времени, то он переходит снова в готовое состояние.

Процесс, который требует дополнительных ресурсов, кроме времени процессора, либо выполнения некоторой операции другим процессом, переводится в заблокированное состояние и ожидает наступления события, которое он потребовал. Как только произошло ожидаемое событие, процесс переводится в готовое состояние.

1 **Условие перехода процесса из подготовительного состояния в готовность.** Выделение ресурсов.
2 **Условие перехода процесса из состояния готовности в активное.** Выделение кванта времени процессора, запуск.
3 **Условие перехода процесса из активного состояния в блокированное.** Ожидание события, освобождения ресурса, конца операции ввода/вывода.
4 **Условие перехода процесса из блокированного состояния в готовое.** Наступление события, освобождения ресурса, завершение ввода/вывода.
5 **Условие перехода процесса из активного состояния в готовое.** Истечение кванта времени. (Выделенное время процессора для выполнение этого процесса завершилось и процессор занят другим процессом).

3) Алгоритм банкира.

Доп. Инфа.

При использовании алгоритма "банкира" подразумевается, что :

- системе заранее известно количество имеющихся ресурсов;
- система знает, сколько ресурсов может максимально потребоваться процессу;
- число пользователей (процессов), обращающихся к ресурсам, известно и постоянно;
- система знает, сколько ресурсов занято в данный момент времени этими процессами (в общем и каждым из них);
- процесс может потребовать ресурсов не больше, чем имеется в системе.

В алгоритме "банкира" введено понятие надежного состояния. Текущее состояние вычислительной машины называется **надежным**, если ОС может обеспечить всем текущим пользователям (процессам) завершение их заданий в течение

конечного времени. Иначе состояние считается **ненадежным**. Надежное состояние — это состояние, при котором общая ситуация с ресурсами такова, что все пользователи имеют возможность со временем завершить свою работу. Ненадежное состояние может со временем привести к тупику.

Система удовлетворяет только те запросы, при которых ее состояние остается надежным, т.е. при запросе ресурса система определяет сможет ли она завершить хотя бы один процесс, после этого выделения.

Пример решения задачи выделения ресурсов по алгоритму "банкира".

Имеем 6 ресурсов и 6 процессов (рис .2.17.). В таблице показано состояние занятости ресурсов на данный момент (1 свободный ресурс).

Процесс	Выделенное число ресурсов	Требуемое число ресурсов
A	2	3
B	1	3
C	0	2
D	1	2
E	1	4
F	0	5

Рис. 2.17

Эта таблица отражает надежное состояние, т.к. существует такая последовательность выделений — освобождений ресурсов, при которой все процессы за конечное время будут завершены.

Недостатки алгоритма банкира:

- Если количество ресурсов большое, то становится достаточно сложно вычислить надежное состояние.
- Достаточно сложно спланировать, какое количество ресурсов может понадобиться системе.
- Число работающих пользователей (процессов) остается постоянным.
- Пользователи должны заранее указывать максимальную потребность в ресурсах, однако, потребность может определяться динамически по мере выполнения процесса.
- Пользователь всегда заказывает ресурсов больше, чем ему может потребоваться.
- Алгоритм требует, чтобы процессы возвращали выделенные им ресурсы в течение некоторого конечного времени. Однако, для систем реального времени требуются более конкретные гарантии. Т.е. в системе должно быть задано максимальное время захвата всех ресурсов процессом (через это время ресурсы должны быть освобождены).

4) Виды приоритетных прерываний

Ответ:

Доп.инфа:

- сигналы (от схем контроллеров, канала i/o, …)

- программы пользователей (i/o, …)

Для ОС более приоритетной явл. 2е

Для системы в общем – 1е

Прерывание (hardware interrupt) – это событие, генерируемое внешним (по отношению к процессору) устройством. Посредством аппаратных прерываний аппаратура либо информирует центральный процессор о том, что произошло какое-либо событие, требующее немедленной реакции (например, пользователь нажал клавишу), либо сообщает о завершении асинхронной операции ввода-вывода.

Всё просто - к примеру, выполняется прерывание низкого приоритета. "Пришло" прерывание высокого
приоритета - прерывается "низкое", выполняется "высокое". После retfie из "высокого" "низкое" продолжает с того места, где его застало "высокое". Это понятно.
А если при выполнении "высокого" "пришло" "низкое" - оно будет выполняться после того как "высокое"
отработает? Вот такая баламуть из "высоких" и "низких". :))

5) Структура дескриптора таблицы страниц.

63 32

База 24-31	G	D	X	U	Предел 16-19	P	DPL	S	Тип Сегмента	A	База 16-23

31 0

База 15-0	15-0 Предел

База – базовый адрес начала сегмента

Предел – кол-во памяти, которое выделено под сегмент

G – бит гранулярности (тип памяти, которую используем)

0 – переход в другое адресное пространство маш. (16 разр.)

1 – память измеряется в страницах

D – тип машины

0 – 16 разрядов

1 - 32 разр.

X – резерв (для 64 разр. маш.)

U – бит для сист. прогр.

p – где находится сегмент (признак)

0 – на диске

1- в памяти

DPL (2-бита)– уровень привелигированости (частота использования)

S – тип сегмента (системный или прикладной)

1 - системный

0 - пользовательский

Тип (3 бита) сегмента – 000 считывание

001 счит. / запись

010 стэк. / счит

011 стэк (чтен\зап)

100 код (только на выполнение)

101 код (выполн, чтение)

110 подчин. сегмент. кода (только активн.)

111 подчин . сегм . кода (активное чтение)

A – тип доступа (активный сегмент или пассивный)