Сформулировать общее (не являющееся определением процессов в Unix) определение процесса.

- Совокупность машинных команд и данных, исполняющаяся в рамках ВС и обладающая правами на владение некоторым набором ресурсов.

Определить понятие длительность цикла оперативной памяти (cycle time, t_{cycle})

- Минимальное время между началом текущего и последующего обращения к памяти.

Время доступа (access time, t_{access})

- Время между запросом на чтение слова из ОП и получением содержимого этого слова.

Определение понятия «взаимное исключение»

- Способ работы с разделяемым ресурсом, при котором в тот момент, когда один из процессов работает с разделяемым ресурсом, все остальные процессы не могут иметь к нему доступ.

Внешнее прерывание.

- Событие, возникающее в компьютере в результате взаимодействия ЦП с внешними устройствами.

Дать полное определение NUMA систем

- MIMD система с общей ОП, имеющая неоднородный доступ в память. Процессорные элементы работают на общем адресном пространстве, но характеристики доступа процессора к ОЗУ зависят от того, в какую часть адресного пространства он обращается.

UMA система

- MIMD система с общей ОП, имеющая однородный доступ в память. Характеристики доступа любого процессорного элемента в любую точку ОЗУ не зависят от конкретного элемента и адреса (все процессоры равноценны относительно доступа к памяти)

Определение виртуального ресурса (устройства)

- Устройство (ресурс), некоторые эксплуатационные характеристики которого (возможно все) реализованы программным образом

Определение семафора Дейкстры

- Семафор представляет собой переменную целочисленного типа S, над которой определены две операции: down(S) (или P(S)) и up(S) (или V(S)).
  down(S) – если значение семафора больше нуля, то уменьшает его на 1. В противном случае процесс блокируется, а операция down считается незавершённой.
  up(S) – увеличивает значение семафора на 1. Если в системе присутствуют процессы, блокированные ранее при выполнении down на этом семафоре, один из них разблокируется с тем, чтобы он завершил выполнение операции down, т.е. вновь уменьшил значение семафора.
  Операции up(S) и down(S) являются атомарными.

Определение понятия «аппарат» виртуальной памяти

- Аппаратно-программные средства компьютера, обеспечивающие преобразование (установление соответствия) программных адресов, используемых в программе с адресами физической памяти, в которой размещена программа во время выполнения.

Что такое инкрементное архивирование файловой системы?

- Единожды создаётся полная или «мастер» копия, все последующие включают только обновлённые файлы.

Что такое «сквозное кеширование»? В чём его преимущество и почему?

- 1. При каждой команде записи в память происходит «сброс» обновлённого блока кеша в ОЗУ.
  2. Нет необходимости в сохранении «вытесненного» блока кеша в ОП.
  3. Данная модель не снижает эффективности работы кеша, т.к. команд чтения из ОЗУ статистически существенно больше, нежели команд записи.

Дать определение тупика (deadlock)

- Ситуация, при которой из-за некорректной организации доступа и разделения ресурсов происходит взаимоблокировка.
- Множество заблокированных процессов, каждый из которых владеет некоторым ресурсом и ожидает ресурса, которым владеет какой-либо другой процесс из этого множества.

Классификация Флинна (перечислить аббревиатуры и названия классов)

- SISD — single instruction, single data stream, SIMD — single instruction, multiple data stream, MISD — multiple instruction, single data stream, MIMD — multiple instruction, multiple data stream.

Что даёт расслоение ОЗУ?

- 1. Сокращение цикла ОП для соседних ячеек
  2. Существенное ускорение работы ОЗУ при обменах блоками

Когда и где наилучшим образом проявляется преимущество ОЗУ с расслоением?

- При использовании кеширования ОЗУ. При блочных обменах между ОЗУ и кеш-буфером.

Минимальные требования к аппаратуре для обеспечения корректного мультипрограммирования.

- 1. Аппарат защиты памяти
  2. Специальный режим ОС (привилегированный режим или режим супервизора)
  3. Аппарат прерываний (как минимум, прерывание по таймеру)

Почему для организации корректного мультипрограммного режима необходимо наличие прерывания по таймеру?

- Обработка (минимизация ущерба) “зацикленных” процессов.

Перечислить состояния, в которых может находиться процесс, обрабатываемый в мультипрограммном режиме.

- 1. Исполнение процессором
  2. Ожидание обмена
  3. Ожидание предоставления процессора для выполнения

Что даёт (с какими целями используется) RAID 0?

- 1. Увеличение суммарного объёма устройства.
  2. Увеличение скорости обмена за счёт «расслоения» информации между независимыми устройствами.

Для каких целей используется RAID 1?

- Зеркалирование

Содержимое каких файлов может размещаться в индексных дескрипторах ФС Unix?

- Файл устройств, файл-ссылка.

Какая качественная информация содержится в коммутаторе драйвера (в записи таблицы драйверов Unix)?

- Адреса стандартных точек входа в драйвер (т.н. коммутатор)

Какое минимальное количество открытых сокетов имеет сервер при реализации модели клиент – сервер средствами аппарата сокетов?

- Количество активных присоединённых клиентов + 1

Чем ограничен предельный объём динамической памяти, который единовременно может быть доступен процессу?

- Объёмом виртуального адресного пространства (грубое ограничение, т.к. следует учесть использованное в процессе виртуальное адресное пространство)

Что описывает содержимое инвертированной таблицы страниц?

- Распределение физических страниц компьютера между виртуальными страницами обрабатываемых процессов.

Что определяет количество записей (размер) инвертированной таблицы страниц?

- Количество записей определяется количеством имеющихся в компьютере физических страниц.

Чему равно количество записей в “прямой” таблице страниц?

- Максимальному количеству виртуальных страниц, допустимых в процессе (максимальному количеству страниц в виртуальном адресном пространстве)

Основное преимущество инвертированной таблицы страниц?

- Нет необходимости перегрузки таблицы при смене обрабатываемых процессов (одна копия таблицы страниц на все обрабатываемые процессы)

В чём состоит основное преимущество использования инвертированных таблиц страниц по сравнению с использованием прямых (не инвертированных) таблиц страниц?

- Нет необходимости перегрузки и сохранения таблиц страниц при смене процессов.

В чём преимущество использования “прямых” (не инвертированных) таблиц страниц по сравнению с использованием инвертированных таблиц страниц?

- Прямая индексация по таблице (быстрый доступ к нужной информации)

Основное преимущество при использовании контроллеров прямого доступа при управлении внешними устройствами (DMA)?

- Исключается непосредственное участие процессора при переносе данных из ВЗУ в ОЗУ (и обратно)

Указать конкретно, в чём состоит основной недостаток использования модели непосредственного управления внешних устройств центральным процессором по сравнению с использованием DMA?

- Избыточная нагрузка на ЦП за счёт обработки потока данных, связанного с обменами.

Основное преимущество протокола TCP по сравнению с протоколом UDP?

- Надёжность при передаче информации. Обеспечивает работоспособность в сетях с недетерминированной надёжностью линий связи

Основное преимущество протокола UDP по сравнению с протоколом TCP?

- Эффективность при передаче данных (объём передаваемой служебной информации существенно меньше)

Основное преимущество использования битовых массивов для учёта свободных блоков ФС?

- Возможность оптимизации размещения данных файлов, минимизирующих фрагментацию данных по устройству (мы можем видеть и использовать свободные области, состоящие из групп свободных блоков)

Преимущества сегментной организации памяти по сравнению со страничной?

- 1. Простая аппаратная реализация (аппаратная таблица сегментов)
  2. Возможность адресной поддержки логической структуры виртуальной памяти процесса (кодовая часть, данные, стек, динамическая память и пр.)

Преимущества страничной организации памяти по сравнению с сегментной?

- 1. Решение проблемы фрагментации памяти.
  2. Оптимальность использования физической памяти – возможность «откачки» значительной части неиспользуемых страниц.

Указать преимущества и недостатки ФС, использующих “маленькие” блоки.

- + Минимизация внутренней фрагментации
  – Увеличение фрагментации данных по устройству
  – Увеличение количества системных вызовов при последовательных обменах – снижение эффективности

Перечислить преимущества и недостатки ФС, использующих “большие” блоки.

- + Эффективность обмена (сокращение количества системных вызовов при последовательных обменах)
  + Сокращение фрагментации данных по устройству
  – Внутренняя фрагментация

Основное преимущество NUMA систем по сравнению с UMA системами?

- Потенциально большая степень параллелизма (в UMA системах существенно ограничено количество процессорных элементов)

Основное достоинство модели инкрементного архивирования ФС?

- Каждая последующая копия архива содержит не все архивируемые файлы, а только те, которые были созданы или изменены с момента создания предыдущей копии.

Основное преимущество алгоритма планирования дисковых обменов N-step-SCAN.

- Разделение очереди на подочереди длины ≤ N запросов каждая (из соображений FIFO). Последовательная обработка очередей. Обрабатываемая очередь не обновляется. Обновление очередей, отличных от обрабатываемой. Борьба с «залипанием» головки (интенсивный обмен с одной и той же дорожкой)

В чём состоит основное преимущество использования легковесных процессов (нитей) по сравнению с использованием полновесных?

- Эффективность организации смены обрабатываемых процессов.

Основное отличие полновесных процессов от легковесных?

- Используют защищённые области ОЗУ.

Принципиальное отличие организации файлов устройств от обыкновенных файлов?

- Содержимое файла располагается в индексном дескрипторе.

Перечислить основные отличия именованных каналов от неименованных.

- 1. Именованные предназначены для взаимодействия произвольных процессов (имеющих доступ к соответствующим файлам), неименованные – только для родственных.
  2. Именованные каналы – разновидность типа файла, могут существовать в системе без процессов, использующих их. Неименованные закрываются после завершения последнего процесса, связанного с ними.

Указать основные принципы организации двухуровневой системы квотирования блоков ФС.

- Жёсткие квоты (лимиты) не превышаются никогда. Гибкие квоты можно превышать, но после этого включается обратный счетчик предупреждений. Пока счетчик > 0, при каждой регистрации пользователя в системе он получает предупреждение, если счетчик = 0, пользователь блокируется.

Основная характеристика алгоритма SSTF – «жадный» алгоритм планирования дисковых обменов?

- На каждом шаге выбирается обмен, требующий минимального перемещения головок диска.

Структура ячейки ОП компьютера.

- Поле тега (служебной информации) + поле машинного слова (поле программно изменяемой информации)

Использование тега ячейки ОП?

- 1. Контроль целостности данных (например, контроль чётности)
  2. Контроль доступа к командам/данным
  3. Контроль доступа к машинным типам данных

Какие процессы в ОС Unix всегда присутствуют в системе (указать pid'ы и название или назначение)

- 0 (соответствует ядру), 1 (init)

Какую конкретную задачу организации доступа к разделяемым ресурсам обеспечивает (Для чего используется) двоичный семафор Дейкстры?

- Взаимное исключение.

Какую модель доступа к ресурсу реализует семафор Дейкстры, у которого предельное значение равно N?

- Единовременный доступ к ресурсу не более N процессов.

Какую модель организации программной системы иллюстрирует задача «спящий парикмахер»?

- Клиент – сервер с ограничением на длину очереди.

Написать общую формулу предельного размера файла Unix в **байтах** при условии, что блок имеет размер ***разм_блока*** байтов, а для адресации блоков используется целое (int).

- $\left(10 + \left(\frac{\text{разм\_блока}}{sizeof(int)}\right) + \left(\frac{\text{разм\_блока}}{sizeof(int)}\right)^2 + \left(\frac{\text{разм\_блока}}{sizeof(int)}\right)^3\right) \times \text{разм\_блока}$

Написать общую формулу предельного размера файла Unix в **блоках** при условии, что блок имеет размер ***разм_блока*** байтов, а для адресации блоков используется целое (int).

- $10 + \left(\frac{\text{разм\_блока}}{sizeof(int)}\right) + \left(\frac{\text{разм\_блока}}{sizeof(int)}\right)^2 + \left(\frac{\text{разм\_блока}}{sizeof(int)}\right)^3$

Что определяет предельно возможное число файлов в ФС s5fs?

- Размер области на диске, выделенный для хранения индексных дескрипторов (или предельное количество индексных дескрипторов)

В ФС s5fs утерян суперблок. Какая минимальная информация необходима для полного автоматического восстановления всех файлов?

- Размер области индексных дескрипторов (или их предельное количество)

Какая информация может размещаться в области блоков ФС fs5?

- Содержимое блоков файлов, списки свободных блоков ФС, ссылки на номера блоков, занимаемых файлами.

Перечислить модельную последовательность действий с ресурсами (данными) файловой системы и файлами Unix при выполнении команды удаления файла. Например, для случая: в текущем каталоге имеется файл NAME и мы выполняем команду rm NAME

- 1. Получаем номер индексного дескриптора из записи каталога
  2. Уменьшаем значение поля Count, содержащее количество ссылок из каталогов к данному ИД
  3. Если значение Count = 0, освобождаем блоки удалённого файла (номера поступают в список свободных блоков), освобождаем данный ИД (он считается свободным, как только счётчик стал равен нулю, пытаемся поместить номер свободного индексного дескриптора в массив суперблока), удаляем запись в текущем каталоге. Стоп
  4. Если значение счётчика больше нуля, удаляем запись в текущем каталоге. Стоп

Может ли ОС (если может, то как) противодействовать зациклившимся процессам в случае, если в системе отсутствует предварительная декларация лимитов времени ЦП для обрабатываемых процессов?

- За счёт переключения с процесса на процесс и изменения приоритетности выделения квантов времени процессам.

Обосновать, что произойдёт с процессом №1 и процессом №2 в результате запуска следующей программы.

```
int main(int argc, char **argv)
{
      int fd[2];
      char c[2];
      pipe(fd);
      if(fork()) { /* процесс №1 */
            close(fd[0]);
            close(fd[1]);
      }
      else { /* процесс №2 */
            read(fd[0],c,1);
      }
      return 0;
}
```

Ответ: процесс №1 завершится, процесс №2 зависнет, т.к. в системе не будет закрыт дескриптор записи в этот канал (дескриптор, который был унаследован процессом №2)

Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.

```
int main()
{
      char c;
      int fd[2],fd2[2];
      pipe(fd);
      pipe(fd2);
      if(fork()==0) {
            write(fd[1],&c,1);
            putchar('b');
            read(fd2[0],&c,1);
            putchar('d');
            exit(0);
      }
      putchar('a');
      read(fd[0],&c,1);
      putchar('c');
      read(fd2[1],&c,1);
      wait(NULL);
      putchar('f');
      return 0;
}
```

Ответ: acbdf либо abcdf либо bacdf

Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.

```
int main()
{
	int fd[2];
	pipe(fd);
	char x[] = "01\n";
	if(fork()) {
		puts(x+1);
		write(fd[1],x,1);
		wait(NULL);
	}
	else {
		write(fd[1],&x[1],1);
		read(fd[0],x,1);
		read(fd[0],x+1,1);
	}
	puts(x);
	return 0;
}
```

Ответ:

1
01
01

либо

1
10
01

Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.

```
int main()
{
	pid_t pid;
	int fd[2];
	int x = 1;
	pipe(fd);
	if( (pid=fork()) > 0 ) {
		read(fd[0],&x,sizeof(int));
		kill(pid,SIGKILL);
		wait(NULL);
	}
	else {
		printf("%d",x);
		x = 2;
		write(fd[1],&x,sizeof(int));
```

```
            x = 3;
      }
      printf("%d",x);
      return 0;
}
```
Ответ: 12 либо 132

Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.

```
int main()
{
      pid_t pid;
      int fd[2];
      int x = 1;
      pipe(fd);
      if( (pid=fork()) > 0 ) {
            close(fd[1]);
            kill(pid,SIGKILL);
            read(fd[0],&x,sizeof(int));
            wait(NULL);
      }
      else {
            x = 2;
            write(fd[1],&x,sizeof(int));
            x = 3;
      }
      printf("%d",x);
      return 0;
}
```
Ответ: 1 либо 2 либо 32

Содержимое файла 1.txt – строка “12345”. Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.

```
int main()
{
      char c = 'a';
      int fd;
      fd = open("1.txt",O_RDONLY);
      if(fork()) {
            int fd2 = open("1.txt",O_RDONLY);
            int fd3 = dup(fd);
            lseek(fd,2,SEEK_SET);
            wait(NULL);
            read(fd2,&c,1); write(1,&c,1);
            read(fd3,&c,1); write(1,&c,1);
```

```
	}
	else {
		read(fd,&c,1); write(1,&c,1);
	}
	return 0;
}
```
Ответ: 113 либо 314

Содержимое файла 1.txt – строка “abcde”. Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.

```
int main()
{
	char c = ‘a’;
	int fd;
	fd = open(“1.txt”,O_RDONLY);
	if(fork()) {
		int fd2 = open(“1.txt”,O_RDONLY);
		int fd3 = dup(fd);
		lseek(fd,2,SEEK_CUR);
		wait(NULL);
		read(fd2,&c,1); write(1,&c,1);
		read(fd3,&c,1); write(1,&c,1);
	}
	else {
		read(fd,&c,1); write(1,&c,1);
	}
	return 0;
}
```
Ответ: aad либо cad

Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.

```
int main()
{
	int fd[2];
	pipe(fd);
	char x;
	if(fork()) {
		putchar(‘a’);
		write(fd[1],&x,1);
		wait(NULL);
		putchar(‘b’);
	}
	else {
		close(fd[1]);
```

```
            kill(getppid(),SIGKILL);
            read(fd[0],&x,1);
            putchar('c');
      }
      return 0;
}
```

Ответ: c либо ac либо acb

Перечислит все варианты вывода на экран. Считается, что все системные вызовы отрабатывают без ошибок, печать на экран происходит без буферизации.

```
#include <unistd.h>
#include <stdio.h>
int main()
{
      int fd[2];
      char *str = "abc";
      char c;
      pipe(fd);
      if(0 == fork()) {
            c = '1';
            write(1,&c,1);
            write(fd[1],str,1);
      } else if(0 == fork()) {
            c = '2';
            write(1,&c,1);
      } else {
            close(fd[1]);
            while(read(fd[0],&c,1) > 0) write(1,&c,1);
            c = '3';
            write(1,&c,1);
      }
      return 0;
}
```

Ответ: 12a3, 21a3, 1a23

1) **В модельной схеме обработки прерываний обсуждалось включение режима блокировки прерывания. На какой стадии обработки прерывания включается данный режим? Почему необходимо включение данного режима (обосновать)?**

   Ответ: Включается на аппаратной стадии. Информация о точке текущего прерывания (адрес прерывания, часть регистров и т.п.) автоматически размещается в специальной регистровой памяти процессора. Если в это время придёт другое прерывание, то информация о точке текущего прерывания будет потеряна, что означает, что мы не сможем продолжить выполнение процесса с прерванного места. Блокировка прерывания необходима для того, чтобы избежать подобных потерь.

2) **Предположим в ОС UNIX System V заблокировано кэширование обменов с блок-ориентированными устройствами. Какое предельное количество непроизводительных обменов (дополнительных обменов) может произойти при чтении блока N файла (считаем, что файл уже открыт). Кратно обосновать.**

   Ответ: Три. Система адресации блоков файлов в индексном дескрипторе использует тройную косвенную адресацию.

3) **Описать алгоритм формирования списка занятых блоков файловой системы fs5.**

   Ответ: Запускаем цикл по области индексных дескрипторов. Считываем очередной ID. Проверяем содержимое поля «ссылки на данный ID каталогов файловой системы». Если это поле равно нулю (это означает, что ID свободен), переходим к следующему ID. В противном случае выбираем и добавляем в список занятых блоков файловой системы номера блоков, занятых данным файлом. Последовательно просматриваем 13 элементов, описывающих адресацию блоков файла (до завершения): номера блоков с прямой адресацией (10 штук), номера блоков, организованных с косвенной адресацией 1, 2 и 3-х уровневой.

4) **В системе используется трёхуровневая таблица страниц процесса. Перечислить все возможные «непроизводительные» действия при обращении к данной таблице во время выполнения процесса.**

   Ответ: Загрузка таблицы второго уровня, загрузка таблицы третьего уровня.

5) **Прокомментировать, что означает требование атомарности операций семафора Дейкстры?**

   Ответ: Это означает, что если операция начала выполняться, то её выполнение не может быть прервано до момента завершения.

6) **Что определяет использование конкетного коммуникационного домена в сокетах (привести примеры)?**

   Ответ: Использование того или иного коммуникационного домена определяет область видимости (возможности использования) сокета (например, AF_UNIX, AF_INET) и правила именования сокетов (например, полное имя файла в случае AF_UNIX или IP адрес + номер порта для AF_INET.

7) **Описать модельную схему установки контрольной точки по адресу Adr в программе средствами ptrace (от чьего имени выполняются действия, в каких состояниях находятся участвующие процессы, какая последовательность действий, сопутствующие данные).**

   Ответ: Родительский процесс устанавливает контрольную точку в сыновнем процессе.

1. Сыновий процесс приостановлен.
2. Считываем (посредством ptrace) команду по адресу Adr сегмента кода сына. Сохраняем пару: Adr, команда.
3. Записываем в сегмент кода (посредством ptrace) сына по адресу Adr команду, выполнение которой вызовет прерывание и возникновение обрабатываемого сигнала. Например, команду, выполняющую деление на ноль.
4. Можем запустить сыновий процесс, например, с точки останова.

1) **Первое поколение компьютеров** - электронно-вакуумные лампы, 1940-1950. Зарождение класса сервисных, управляющих программ. Зарождение языков программирования. Однопользовательский, персональный режим.

2) **Второе поколение компьютеров** - полупроводниковые приборы: диоды и транзисторы, 1950-1960, пакетная обработка заданий, мультипрограммирование, языки управления заданиями, файловые системы, виртуальные устройства, операционные системы.

3) **Третье поколение компьютеров** - интегральные схемы малой и средней интеграции, 1960-1970, аппаратная унификация узлов и устройств, создание семейств компьютеров, унификация компонентов программного обеспечения.

4) **Компьютеры четвертого поколения** - большие и сверхбольшие интегральные схемы, 1970-х – настоящее время. «Дружественность» пользовательских интерфейсов, сетевые технологии, безопасность хранения и передачи данных.

5) **Вычислительная система** - совокупность аппаратных и программных средств, функционирующих в единой системе и предназначенных для решения задач определенного класса.

6) **Уровни вычислительной системы** снизу-вверх: аппаратный, уровень физических устройств, уровень логических устройств, системы программирования, прикладные системы.

7) **Драйвер физического устройства** – программа, основанная на использовании команд управления конкретного физического устройства и предназначенная для организации работы с данным устройством.

8) **Логическое/виртуальное устройство (ресурс)** – устройство/ресурс, некоторые эксплутационные характеристики которого (возможно все) реализованы программным образом.

9) **Драйвер логического/виртуального ресурса** - программа, обеспечивающая существование и использование соответствующего ресурса.

10) **Ресурсы вычислительной системы** - совокупность всех физических и виртуальных ресурсов.

11) **Операционная система** - это комплекс программ, обеспечивающий управление ресурсами вычислительной системы.

12) **Система программирование** – это комплекс программ, обеспечивающий поддержание жизненного цикла программы в вычислительной системе.

13) **Этапы жизни программы**: проектирование, кодирование, тестирование, отладка, изготовление.

14) **Проектирование** - исследование задачи, исследование характеристик объектной среды (как объектная среда будет связана с нашей системой).

15) **Объектная среда** – это та ВС, в рамках которой продукт будет функционировать.

16) **Инструментальная среда** – это ВС, которая будет использована для разработки программ.

17) **Построение кода на основании спецификаций при использовании языков программирования, трансляторов, средств для использования библиотек и средств для разработки программных продуктов.** Результатом этапа кодирования являются исполняемые модули, объектные модули, исходные тексты программ и библиотеки.

18) **Тестирование** – это проверка спецификаций функционирования программы на некоторых наборах входных данных.

19) **Отладка** – процесс поиска, анализа и исправления зафиксированных при тестировании и эксплуатации ошибок.
20) **Современные разработки программного обеспечения**: каскадная модель, каскадно-иттерационная, спиральная.

21) **Прототип** - программа, частично реализующая функциональность и внешний интерфейс разрабатываемой системы.

22) **Виртуальная машина** - программно реализованные апаратные средства. Спецификация некоторой вычислительной среды

23) **Прикладная система** – программная система, ориентированная на решение или автоматизацию решения задач из конкретной предметной области.

24) **Принципы работы компьютера Фон Неймана**: принцип двоичного кодирования, принцип хранимой программы, принцип программного управления.

25) **Центральный процессор (ЦП**) – компонент компьютера, обеспечивающий выполнение программ. Программы, выполняемые в рамках процессора, координируют работу ОЗУ и внешних устройств.

26) **Оперативно запоминающее устройство (ОЗУ)** – устройство хранения данных, в котором размещается исполняемая в данный момент программа и из которого выбираются команды и данные этой программы.

27) **Внешние устройства** – программно управляемые устройства, входящие в состав компьютера.

28) **Арифметически-логическое устройство (АЛУ)** - устройство, реализующее команды, которые подразумевают обработку данных.

29) **Устройство управления (УУ)** - обеспечивает последовательный выбор команд, которые необходимо выполнить программе, их контроль, дешифровку и, в зависимости от типа команды, последующую обработку.

30) **ОЗУ состоит** из ячеек памяти, которые состоят из **тегов** (поле служебной информации) и **машинных слов (поле программно изменяемой информации).**

31) **Производительность оперативной памяти** - скорость доступа процессора к данным, размещенным в ОЗУ.

32) **Время доступа (access time- taccess)** - время между запросом на чтение слова из оперативной памяти и получением содержимого этого слова.

33) **Длительность цикла памяти (cycle time - tcycle)** - минимальное время между началом текущего и последующего обращения к памяти. (tcycle>taccess)

34) **Расслоение ОЗУ** – один из аппаратных путей решения проблемы дисбаланса в скорости доступа к данным, размещенным в ОЗУ и производительностью ЦП.

35) **Регистровая память** – совокупность устройств памяти ЦП ,предназначенных для временного хранения операндов, информации, результатов операций. (Регистр Адреса (РА), Регистр Результата (РР),Слово – состояние процессора (ССП или PSW), Регистры внешних устройств (РВУ) , Регистр указатель стека (РУС).

36) **КЭШ память** - (Аппаратное решение) буферизация работы процессора с оперативной памятью.

37) **Стратегии вытеснения КЭШ памяти** — случайное, наименее популярного.

38) **Процесс вытеснения** — сквозное, с обратной связью.

39) **Прерывание** - событие в компьютере, при возникновении которого в процессоре происходит предопределенная последовательность действий.

40) **Внутреннее прерывание** - инициируются схемами контроля работы процессора.

41) **Внешнее прерывание** - события, возникающие в компьютере в результате взаимодействия.

42) **Внешние запоминающие устройства (ВЗУ)**– устройства, предназначенные для хранения данных и программ.

43) **Устройства ввода и отображения информации** – осуществляют ввод из вне некоторой информации и отображение ее в виде некоторых результирующих данных.

44) **Устройства приема и передачи данных** используются для получения данных с других компьютеров, «из вне».

45) **ВЗУ Последовательного доступа**: Магнитная лента

46) **ВЗУ Прямого доступа**: магнитные диски, магнитный барабан, магнито-электронные ВЗУ прямого доступа.

47) **Поток информации** - программно заданная информация о том, что необходимо прочесть или записать или переместить данные из одного места в другое.

48) **Поток данных** -непосредственно ответ на управляющее воздействие и связанное с этим ответом перемещение данных от внешнего устройства в ОП или в ЦП.

49) **DMA (direct memory access)** – устройство, позволяющее обращаться к ОЗУ из ВЗУ без посредника.

50) **Иерархия памяти**: регистры общего назначения, КЭШ 1-го уровня, КЭШ 2-го уровня, ОЗУ, внешнего запоминающего устройство с внутренней КЭШ-буферизацией, внешнего запоминающего устройство без внутренней КЭШ-буферизации, внешнего запоминающего устройство длительного хранения.

51) **Мультипрограммный режим** - режим при котором возможна организация переключения выполнения с одной программы на другую.

52) **Аппаратные средства компьютера, необходимые для поддержания мультипрограммного режима**: аппарат прерываний (хотя бы прерывание по таймеру), аппарат защиты памяти, специальный режим ОС (привилегированный).

53) **Регистровые окна** – один из способов решения проблемы вложенных процедур.

54) **Аппарат виртуальной памяти** – аппаратные средства компьютера, обеспечивающие преобразование (установление соответствия) программных адресов, используемых в программе адресам физической памяти в которой размещена программа при выполнении.

55) **Базирование адресов** – реализация одной из моделей аппарата виртуальной памяти. При базировании выделяется регистр, в котором будет храниться адрес, начиная с которого размещается программа.

56) **Страничная организация** — аппаратная организация памяти, при которой все пространство делится на фрагменты одного размера (обычно 2^K)

57) **Виртуальное адресное пространство** – множество виртуальных страниц, доступных для использования в программе. Количество виртуальных страниц определяется размером поля «номер виртуальной страницы» в адресе.

58) **Физическое адресное пространство** – оперативная память, подключенная к данному компьютеру. Физическая память может иметь произвольный размер (число физических страниц может быть меньше, больше или равно числу виртуальных страниц).

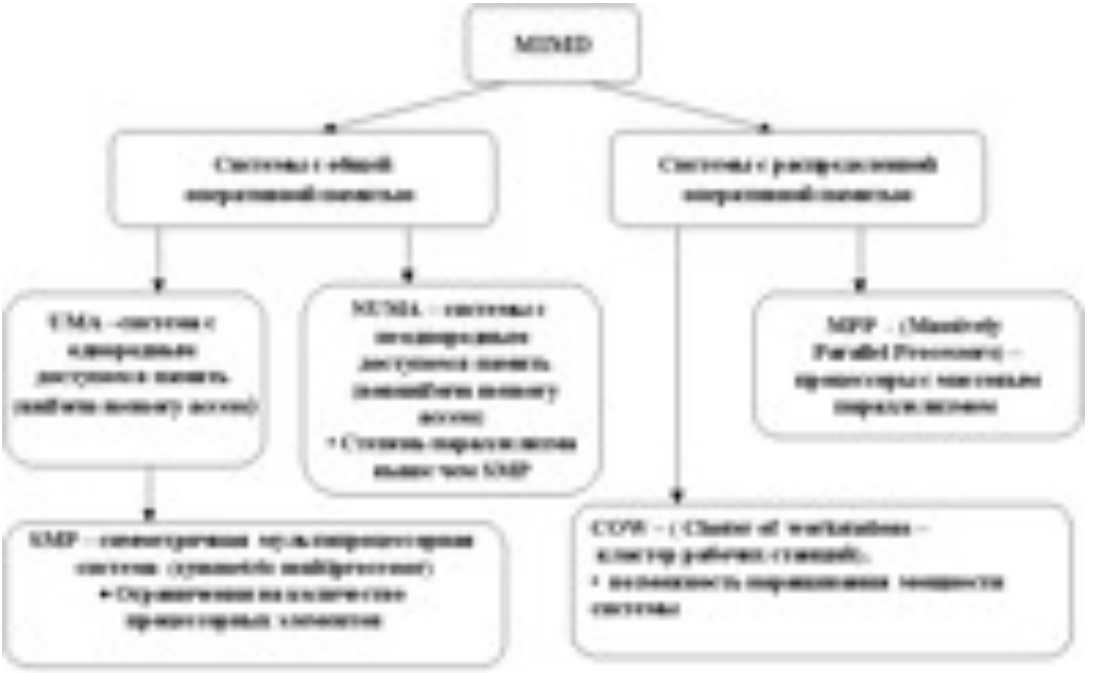
59) **Классификация Флинна**: поток управляющей информации – собственно команд (инструкций), и поток данных. Считаем потоки данных и команд независимыми (условно). Рассмотриваем все возможные комбинации.

60) **ОКОД (SISD – single instruction (одиночный поток команд), single data stream, (одиночный поток данных))** Традиционные компьютеры, которые мы называем однопроцессорными.

61) **ОКМД(SIMD – single instruction(одиночный поток команд), multiple data stream(множественный поток данных))** Для каждой команды порция данных (векторная или матричная обработка данных)
62) **МКОД (MISD – multiple instruction(множественный поток команд), single data stream(одиночный поток данных))** – это вырожденная категория, считается, что ее нет. (Но есть вырожденные случаи — обработка графики)

63) **МКМД** (MIMD - multiple instruction(множественный поток команд), multiple data stream(множественный поток данных))

Хм, ну это тут лучше всего будет смотреться, определения на картинке:



64) **Гетерогенные** – системы объединяющие кластеры разных мощностей. Преимущества кластеров: 1) относительная дешевизна; 2) способность к расширению, увеличению мощностей.

65) **Терминальный комплекс** – это многомашинная ассоциация предназначенная для организации массового доступа удаленных и локальных пользователей к ресурсам некоторой вычислительной системы.

66) **Основная вычислительная система** – система, массовый доступ к ресурсам которой обеспечивается терминальным комплексом.

67) **Локальные мультиплексоры** – аппаратные комплексы, предназначенные для осуществление связи и взаимодействия вычислительной системы с несколькими устройствами через один канал ввода/вывода.

68) **Локальные терминалы** – оконечные устройства, используемые для взаимодействия пользователей с вычислительной системой.

69) **Модемы** – устройства, предназначенные для организации взаимодействия вычислительной системы с удаленными терминалами с использованием телефонной сети. В функцию модема входит преобразование информации из дискретного, цифрового представления.

70) **Удаленные терминалы** – терминалы, имеющие доступ к вычислительной системе с использованием телефонных линий связи и модемов.

71) **Удаленные мультиплексоры** – мультиплексоры, подключенные к вычислительной системе с использованием телефонных линий связи и модемов.

72) **Виды каналов**: коммутируемые (каждый раз новый), выделенные(зарезервированный).

73) **Симплексные каналы** - каналы, по которым передача информации ведется в одном направлении

74) **Дуплексные каналы** - каналы, которые обеспечивают одновременную передачу информации в двух направлениях (например, телефонный разговор, мы одновременно можем и говорить и слушать).
75) **Полудуплексные каналы** - каналы, которые обеспечивают передачу информации в двух направлениях, но в каждый момент времени только в одну сторону (подобно рации).

76) **Компьютерная сеть** – объединение компьютеров (или вычислительных систем), взаимодействующих через коммуникационную среду.

77) **Коммуникационная среда** – каналы и средства передачи данных.

78) **Абонентские машины (хосты)** - обеспечивают обмен содержательной информацией работы с

пользователями.

79) **OSI** – системы открытых интерфейсов.

80) **Уровни взаимодействия компьтеров**: физический уровень, канальный уровень, сетевой уровень, транспортный уровень (уровень логического канала), сеансовый уровень, представительский уровень, прикладной уровень.

81) **Протокол** – формальное описание сообщений и правил, по которым сетевые устройства (вычислительные системы) осуществляют обмен информацией. Или правила взаимодействия одноименных уровней.

82) **Интерфейс** – правила взаимодействия вышестоящего уровня с нижестоящим.

83) **Служба или сервис** – набор операций, предоставляемых нижестоящим уровнем вышестоящему.

84) **Стек протоколов** – перечень разноуровневых протоколов, реализованных в системе

85) **Сообщение** — логически целостная порция данных, имеющая произвольный размер.

86) **Сеть коммутации каналов** — обеспечивает выделение коммутации связи на весь сеанс связи.

87) **Сеть коммутации пакетов** — обеспечивает выделение коммутации связи на передачу одного пакета.

88) **IP адрес** - последовательностью четырех байтов. В адресе кодируется уникальный номер сети, а также номер компьютера (сетевого устройства в сети).

89) **Пакет** – это блок данных, который передаётся вместе с информацией, необходимой для его корректной доставки. Каждый пакет перемещается по сети независимо от остальных.

90) **Дейтаграмма** – это пакет протокола IP. Контрольная информация занимает первые пять или шесть 32-битных слов дейтаграммы. Это её заголовок (header). По умолчанию, его длина равна пяти словам, шестое является дополнительным. Для указания точной длины заголовка в нём есть специальное поле – длина заголовка (IHL, Internal Header Length).

91) **Шлюз** – устройство, передающее пакеты между различными сетями.

92) **Маршрутизация** – процесс выбора шлюза или маршрутизатора

93) **Протокол контроля передачи (TCP, Transmission Control Protocol)** - обеспечивает надежную доставку данных с обнаружением и исправлением ошибок и с установлением логического соединения.

94) **Протокол пользовательских дейтаграмм (UDP, User Datagram Protocol)** - отправляет пакеты с данными, «не заботясь» об их доставке.

95) **Протоколы, опирающиеся на TCP** (TELNET (Network Terminal Protocol), FTP (File Transfer Protocol), SMTP (Simple Mail Transfer Protocol),

96) **Протоколы, опирающиеся на UDP** , DNS (Domain Name Service), RIP (Routing Information Protocol), NFS (Network File System)

97) **Сетевая ОС** – физическую сеть в которой подключенные компьютеры взаимодействуют с помощью протоколов, сетевая ОС предоставляет пользователям распределенные прикладные приложения

98) **Распределенная ОС** — ОС, функционирующая на многопроцессорном или многомашинном комплексе, в котором на каждом из узлов функционирует свое ядро, а также система, обеспечивающая распределение возможностей (ресурсов) ОС.

99) **Процесс** – это совокупность машинных команд и данных, обрабатываемых в рамках ВС и обладающая правами на владение некоторым набором ресурсов. (Вообще определений много)

100) Существуют ресурсы, которые монопольно принадлежат данному процессу, и **разделяемые ресурсы**, которые принадлежат одновременно 2 и более процессам.

101) **Выделения ресурсов процессу**: предварительная декларация использования тех или иных ресурсов, динамическое пополнение списка принадлежащих процессу ресурсов по ходу.

102)**Любая ОС должна удовлетворять набору свойств**: надежность, обеспечение защиты, эффективность, предсказуемость.

103)**Резидентная** – постоянно находящаяся в памяти

104)**Ядро (kernel)** – резидентная часть ОС, работающая в режиме супервизора. («обычно» работает в режиме физической адресации).

105)**Системный вызов** - средство ОС, обеспечивающее возможность процессов обращаться к ОС за теми или иными функциями.

106)**Монолитное ядро** – ядро, которое включает в себя все возможности операционной системы, запускаются как единый процесс.

107)Существует стационарное **микроядро**, которое обеспечивает минимальные функции ОС (на нем все держится)

108)**Логические функции ОС**: управление процессами, управление ОП, планирование, управление устройствами и ФС

109)**Буфер ввода процессов** -область на внешней памяти, где аккумулируются все процессы, которые еще не начали выполняться.

110)**Буфер обрабатываемых процессов** - область памяти, где хранятся данные процессов, которые начали обрабатываться в мультипрограммном режиме.

111)**Пакет программ** – совокупность программ, для выполнения каждого из которых требуется некоторое время работы процессора.

112)**Квант времени ЦП** – некоторый фиксированный ОС промежуток времени работы ЦП

113)**Системы реального времени** - являются специализированными системами в которых все функции планирования ориентированы на обработку некоторых событий за время, не превосходящее некоторого предельного значение

114)**Системы с разделением времени** – системы, в которых на выполенние каждого процесса отводится определенный промежуток (квант) процессорного времени.

115)**Считывание информации, алгоритмы: простейшая модель** – случайная выборка из очереди, FIFO, LIFO, SSTF – жадный, Приоритетный алгоритм (RPI) – это алгоритм, когда последовательность обменов (очередь) имеет характеристику приоритетов, SCAN, C-SCAN, N-step-SCAN.

116)**RAID система** представляет собой набор независимых дисков, которые рассматриваются ОС как единое дисковое устройство, где данные представляются в виде последовательности записей, которые называются полосы.

117)**Специальные файлы устройств** - в системе Unix единый интерфейс организации взаимодействия с внешними устройствами

118)**Файлы байториентированных устройств** (драйверы обеспечивают возможность побайтного обмена данными и, обычно, не используют централизованной внутрисистемной кэш-буферизации);

119)**Файлы блокориентированных устройств** (обмен с данными устройствами осуществляется фиксированными блоками данных, обмен осуществляется с использованием специального внутрисистемного буферного кэша).

120)**Индексный дескриптор** файла устройства содержит: тип файла устройства – байториентированный или блокориентированный, «старший номер» (major number) устройства - номер драйвера в соответствующей таблице драйверов устройств, «младший номер» (minor number) устройства – служебная информация, передающаяся драйверу устройства.

121)**bdevsw** – таблица драйверов блокориентированных устройств.

122)**cdevsw** - таблица байториентированных устройств.

123)**Существует два, традиционных способа включения драйверов новых устройств в систему**: путем «жесткого», статического встраивания драйвера в код ядра, требующего перекомпиляцию исходных текстов ядра или пересборку объектных модулей ядра и за счет динамического включения драйвера в систему.

124)**Таблица индексных дескрипторов открытых файлов**: для каждого открытого в рамках системы файла формируется запись в таблице ТИДОФ, содержащая: копии индексного дескриптора (ИД) открытого файла, кратность - счетчик открытых в системе файлов, связанных с данным ИД.

125)**Таблица файлов**: таблица файлов содержит сведения о всех файловых дескрипторах открытых в системе файлов.

126)**Таблица открытых файлов**: С каждым процессом связана таблица открытых файлов (ТОФ). Номер записи в данной таблице есть номер ФД, который может использоваться в процессе. Каждая строка этой таблицы имеет ссылку на соответствующую строку ТФ.

127)**Организация буферизации при обмене**. В RAM организуется пул буферов, где каждый буфер имеет размер в один блок. Каждый из этих блоков может быть ассоциирован с драйвером одного из физических блок

128)**SYNC** - по этой пользовательской команде осуществляется сброс данных на диск.

129)**Основные задачи при организации работы с ОП**: контроль состояния каждой единицы памяти (свободна/распределена), сратегия распределения памяти, выделение памяти, стратегия освобождения памяти.

130)**Стратегии и методы управления**: одиночное непрерывное распределение, распределение разделами, распределение перемещаемыми разделами, страничное распределение, сегментное распределение, сегменто-страничное распределение.

131)**TLB (Translation Lookaside Buffer)** – Буфер быстрого преобразования адресов.

132)**ХЭШ – функция** берет номер виртуальной страницы и по этому номеру виртуальной страницы имеется некоторая функция, которая определяет номер записи хэш-таблицы.

133)Алгоритм NRU (Not Recently Used – не использовавшийся в последнее время)
Используются биты статуса страницы. R – обращение, M – модификация. Устанавливаются аппаратно при обращении или модификации.

134)**Алгоритм FIFO** – first in, first out.

135)**Алгоритм LIFO** – last in, first out.

136)**Алгоритм LRU** (Least Recently Used – «менее недавно» - наиболее давно используемая страница)

137)**Алгоритм NFU** (Not Frequently Used – редко использовавшаяся страница)
138)**Файловая система (ФС)** - часть операционной системы, представляющая собой совокупность организованных наборов данных, хранящихся на внешних запоминающих устройствах, и программных средств, гарантирующих именованный доступ к этим данным и их защиту

139)Данные называются файлами, их имена - именами файлов.

140)**Правило работы с файлами**: открытие, работа с файлом, закрытие.

141)**Файловый дескриптор** – системная структура данных, содержащая информацию о актуальном состоянии «открытого» файла.

142)**Каталог** – компонент файловой системы, содержащий информацию о содержащихся в файловой системе файлах. (Каталог — это файл)

143)**Суперблок** – блок ФС, в котором находится информация о настройках ФС и актуальном состоянии ФС (информация о свободных блоках, данных, которые содержат каталоги.)

144)**Блок** – порция данных, фиксированного размера, в рамках которого идет обмен данными с устройством.

145)**Таблица размещения файловой системы** - таблица, в которой количество строк соответствует количеству блоков, в кот. i-ая строка соотв. i-ому блоку файловой системы.

146)**Индексные узлы или индексные дескрипторы** — организованное файловой системой компактное хранение информации о размещении блоков файлов в специальной структуре.

147)**“Жесткая” связь**: Есть содержимое файла, есть атрибут файла, и одним из полей атрибутов является количество имен у этого файла, и есть произвольное количество имен, которые как-то распределены по каталогам ФС.

148) **“Символическая” связь** есть файл с именем $Name_2$, этому имени соответствуют атрибуты и соответствует содержимое, и есть специальный файл $Name_1$, который ссылается на имя $Name_2$.

149)**Физическая архивация** («один в один» или интеллектуальная физическия архивация (копируются только использованные блоки файловой системы)

150)**Логическая архивация** – копирование файлов (а не блоков), модифицированных после заданной даты.

151)**Файл Unix** – это специальным образом именованный набор данных, размещенный в файловой системе.

152)**Обычный файл (regular file)** – традиционный тип файла, содержащий данные пользователя. Интерпретация содержимого файла производится программой, обрабатывающей файл.

153)**Каталог (directory)** – специальный файл, обеспечивающий иерархическую организацию файловой системы. С каталогом ассоциируются все файлы, которые принадлежат данному каталогу.

154)**Специальный файл устройств (special device file)** – система позволяет ассоциировать внешние устройства с драйверами и предоставляет доступ к внешним устройствам, согласно общим интерфейсам работы с файлами.

155)**Именованный канал (named pipe)** – специальная разновидность файлов, позволяющая организовывать передачу данных между взаимодействующими процессами;

156)**Ссылка (link)** – позволяет создавать дополнительные ссылки к содержимому файла из различных точек файловой системы; Они могут нарушать древовидность организации ФС.

157)**Сокет (socket)** – средство взаимодействия процессов в пределах сети ЭВМ.

158)**Корневой каталог** / является основой любой файловой системы ОС UNIX. Все остальные файлы и каталоги располагаются в рамках структуры, порожденной корневым каталогом, независимо от их физического положения на диске.

159)**/unix** - файл загрузки ядра ОС.

160)**/bin** - файлы, реализующие общедоступные команды системы.

161)**/etc** - в этом каталоге находятся файлы, определяющие настройки системы

162)**/tmp** - каталог для хранения временных системных файлов. При перезагрузке системы не гарантируется сохранение его содержимого.

163)**/mnt** - каталог, к которому осуществляется монтирование дополнительных физических файловых систем для получения единого дерева логической файловой системы. Заметим, что это лишь соглашение, в общем случае можно примонтировать к любому каталогу.

164)**/dev** - каталог содержит специальные файлы устройств, с которыми ассоциированы драйверы устройств.

165)**/lib** - здесь находятся библиотечные файлы языка Си и других языков программирования.

166)**/usr** - размещается вся информация, связанная с обеспечением работы пользователей. Здесь также имеется подкаталог, содержащий часть библиотечных файлов (/usr/lib), подкаталог /usr/users (или /usr/home), который становится текущим при входе пользователя в систему, подкаталог, где находятся дополнительные команды (/usr/bin), подкаталог, содержащий файлы заголовков (/usr/include), в котором,

в свою очередь, подкаталог, содержащий include-файлы, характеризующие работу системы (например, signal.h - интерпретация сигналов).

167)**Индексный дескриптор в ОС UNIX** – это специальная структура данных файловой системы, которая ставится во взаимно однозначное соответствие с каждым файлом.

168)**Процесс** - совокупность машинных команд и данных,которая исполняется в рамках вычислительной системы и обладает правами на владение некоторым набором ресурсов.

169)**Разделяемые ресурсы** - ресурсы разделяемые между процессами

170)**Буфер ввода процесса (БВП)** — пространство, в котором размещаются и хранятся сформированные процессы от момента их образования, до момента начала выполнения.

171)**Буфер обрабатываемых процессов (БОП)** — буфер, где размещаются процессы, работающие в мультипрограммном режиме.

172)**«Полновесные процессы»** - это процессы, выполняющиеся внутри защищенных участков памяти операционной системы, то есть имеющие собственные виртуальные адресные пространства для статических и динамических данных

173)**Легковесные процессы**, называемые еще как нити или сопрограммы, не имеют собственных защищенных областей памяти.

174) **Контекст процесса** - совокупность данных, характеризующих актуальное состояние процесса.

175)**Процесс в ОС Unix** – объект, зарегистрированный в таблице процессов Unix или объект, порожденный системным вызовом fork()

176)**Идентификатором процесса (PID)** - уникальным имя процесса. PID – целое число от 0 до некоторого предельного значения, определяющего максимальное число процессов (ресурс данной ОС), существующих в системе одновременно.

177)**Сегмент кода** содержит машинные команды и неизменяемые константы соответствующей процессу программы.

178)**Сегмент данных** – содержит данные, динамически изменяемые в ходе выполнения кода процесса.

179)**Системный вызов** – специальная функция, позволяющая процессу обращаться к ядру ОС за выполнением тех или иных действий.

180)**Семейство системных вызовов exec()** - заменяет тело вызывающего процесса, после чего данный процесс начинает выполнять другую программу. Управление передается на точку ее входа. Возврат к первоначальной программе происходит только в случае ошибки при обращении к exec() , т.е. если фактической замены тела процесса не произошло.

181)**Квант времени** – непрерывный период процессорного времени.

182)**Приоритет процесса** – числовое значение, показывающее степень привилегированности процесса при использовании ресурсов ВС (в частности, времени ЦП).

183)**Невытесняющая стратегия** - если величина кванта не ограничена

184)**Вытесняющая стратегия** - величина кванта ограничена.

185)**Простой круговорот (RR – round robin**) не использует никакой статистической или динамической информации о приоритетах.

186)При **круговороте со смещением** каждому процессу соответствует своя длина кванта, пропорциональная его приоритету.

187)**«Эгоистический» круговорот**. Если параметры A и B : 0<=B<A. Процесс, войдя в систему ждет пока его приоритет не достигнет приоритета работающих процессов, а далее выполняется в круговороте. Приоритет выполняемых процессов увеличивается с коэффициентом B<A, следовательно, ожидающие процессы их догонят. При B=0 «эгоистический» круговорот практически сводится к простому

188)**Область свопинга** - специально выделенное системой пространство внешней памяти

189)**Параллельные процессы** - Процессы, выполнение которых хотя бы частично перекрывается по времени

190)**Независимые процессы** – процессы, использующие независимое множество ресурсов и на результат работы такого процесса не влияет работа независимого от него процесса.

191)**Взаимодействующие процессы** совместно используют ресурсы, и выполнение одного может оказывать влияние на результат другого.

192)**Критические ресурсы** — разделяемые ресурсы, которые должны быть доступны в текущий момент времени только одному процессу.

193)**Критическая секция, или критический интервал** - часть программы (фактически набор операций), в которой осуществляется работа с критическим ресурсом.

194)**Взаимное исключение** – т.е. такой способ работы с разделяемым ресурсом, при котором постулируется, что в тот момент, когда один из процессов работает с разделяемым ресурсом, все остальные процессы не могут иметь к нему доступ.

195)**Тупики (deadlocks)** - ситуации в которой конкурирующие за критический ресурс процессы безвозвратно блокируются.

196)**Блокирование** — ситуация, когда доступ одного из процессов к разделяемому ресурсу не обеспечивается из-за активности других, более приоритетных процессов.

197)**Семафоры** – это низкоуровневые средства синхронизации, для корректной практической реализации которых необходимо наличие специальных, атомарных семафорных машинных команд.

198)**Семаоры Дейкстры** — формальная модель, предложенная голландцем Дейкстрой, которая основывается на следующем предположении: имеется тип данных, именуемой семафором (только 2 значения). Над семафором определенны 2 операции: увеличить, уменьшить.

199) **Монитор Хоара** — совокупность процедур и структур данных, объединенных в программный модуль специального вида.

200)**Сигнал** – средство уведомления процесса о наступлении некоторого события в системе.
Инициаторы посылки сигнала - другой процесс или ОС.

201) **Неименованный канал**- область на диске, к которой не возможен доступ по имени, а только с помощью двух дискрипторов с ней ассоциированных.

202)**Именованный канал** - расширение понятия конвейера в Unix.

203)**Система IPC** – альтернатива именованным каналам. Позволяет организовывать работу именованных каналов в произвольные моменты времени.

204)**Очередь сообщений** представляет собой некое хранилище типизированных сообщений, организованное по принципу FIFO.

205)**Механизм разделяемой памяти** позволяет нескольким процессам получить отображение некоторых страниц из своей виртуальной памяти на общую область физической памяти.

206)**Семафоры** представляют собой одну из форм IPC и используются для синхронизации доступа нескольких процессов к разделяемым ресурсам, т.е. фактически они разрешают или запрещают процессу использование разделяемого ресурса.

207)**Сокет** – программный интерфейс для обеспечения информационного обмена между процессами.
Удачи!