Контрольные вопросы

1.Что такое внутренние и внешние команды Shell-интерпретатора? Приведите примеры.

Внутренняя команда – команда, выполняющаяся непосредственно интерпретатором без порождения процесса, она встроена в Bash. Для выполнения не нужно искать заданный путь в переменной PATH. Используют для: Для производительности (внешние запускаются в дочернем процессе) Прямой доступ к внутренним структурам интерпретатора
Примеры: source, cd, fg и т. д. **cd [справочник]** Объявить указанный справочник текущим. Если параметр не задан, в качестве имени справочника используется значение макропеременной HOME. Синонимом команды cd является команда chdir. **Команда fg** в Linux используется для перевода фонового задания на передний план.

Внешняя команда – команда, не встроенная в оболочку. При выполнении оболочка ищет путь, а также запускается новый процесс. Находятся обычно в /usr/bin/
Примеры: ls, cat и т.д. ls - список содержимого каталога cat – покажет содержимое данного файла

2. Какие существуют средства группирования команд и перенаправления ввода-вывода? Приведите примеры использования.

Разделитель команд	**Порядок исполнения**
;	Первая команда, затем вторая команда
&&	Вторая команда будет выполнена только при успешном завершении первой команды
||	Вторая команда будет выполнена только в случае невыполнения первой команды.
|	Выполняется первая команда и ее выходные данные подаются на вход второй команды
Примеры использования	
cmd1 arg ...; …; cmdN arg ...	последовательное выполнение команд;
cmd1 arg ...&...& cmdN arg ...	асинхронное выполнение команд; (Следует перейти к обработке следующей команды без ожидания окончания трансляции)
cmd1 arg ... && cmd2 arg ...	зависимость последующей команды от предыдущей таким образом, что последующая команда выполняется, если предыдущая выдала нулевое значение;
cmd1 arg ... || cmd2 arg ...	зависимость последующей команды от предыдущей таким образом, что последующая команда выполняется, если предыдущая выдала ненулевое значение;
Перенаправление ввода-вывода	
cmd > file	стандартный вывод направлен в файл file;
cmd >> file	стандартный вывод направлен в конец файла file; (переназначить вывод с добавлением данных к уже существующим)
cmd < file	стандартный ввод выполняется из файла file;
cmd1 | cmd2	конвейер команд, в котором стандартный вывод команды cmd1 направлен на стандартный вход команды cmd2.

3. В чем сущность конвейера команд? Приведите примеры использования.

К каждой программе, запускаемой в командной строке, по умолчанию подключено три потока данных:

STDIN (0)	стандартный поток ввода (данные, загружаемые в программу).
STDOUT (1)	стандартный поток вывода (данные, которые выводит программа). По умолчанию — терминал.
STDERR (2)	стандартный поток вывода диагностических и отладочных сообщений (например, сообщениях об ошибках). По умолчанию — терминал.

Конвейеры и перенаправления — это инструменты, которые позволяют связывать потоки между программами и файлами нужным способом.

Конвейеризация – отправка данных из одной программы в другую.

Для конвейеризации существует специальный оператор — **|**

Как он работает: вывод из программы слева от оператора передается в качестве ввода в программу справа от оператора.

В примере мы выводим только первые три файла директории с помощью оператора конвейеризации.	1. **user@bash: ls** 2. barry.txt bob example.png firstfile foo1 myoutput video.mpeg 3. **user@bash: ls | head -3** 4. barry.txt 5. bob 6. example.png 7. **user@bash:**

Еще пример: who| wc число работающих Unix-пользователей

4. Как выполняется подстановка результатов выполнения команд?

Shell-интерпретатор дает возможность выполнять подстановку результатов выполнения команд в Shell-программах. Если команда заключена в одиночные обратные кавычки, то интерпретатор Shell выполняет эту команду и подставляет вместо нее полученный результат.

Если в командной строке встретилась цепочка символов, заключенная в обратные кавычки (`), она интерпретируется как команда, стандартный вывод которой подставляется вместо упомянутой конструкции.

Пример:
Пусть файл filelist содержит список имен других файлов и над совокупностью последних требуется проделать некоторое действие.

Если напишем так ... `cat filelist` ..., то cat будет «ссылкой/указателем» на все файлы списка.

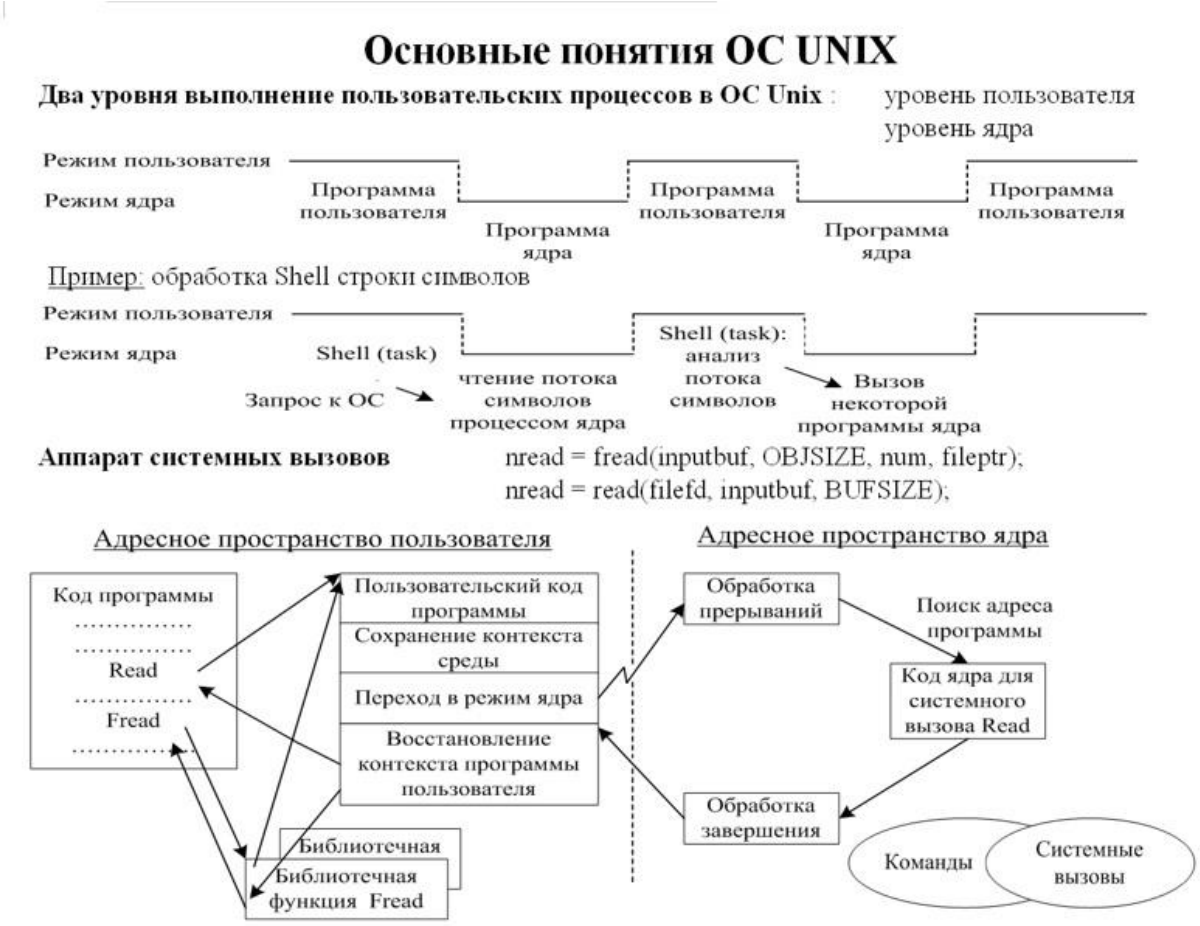
А если напишем так: ls -l `cat filelist` ,то будет выдана информация о файлах (подстановка результатов выполнения команды)

5. В каком режиме выполняется интерпретатор команд Shell?

В интерпретаторе shell существует два режима работы: выполнение команд из командной строки и последовательное выполнение всех команд из файла.

6. Кем и в каком режиме осуществляется чтение потока символов с терминала интерпретатором Shell?

Shell читает вводимые символы с терминала и, если нет ошибок, то передает строку программе обработчику. Программа обрабатывает введенные данные и возвращает результат Shelly, а он выводит результат на терминал.



7. Что представляет собой суперблок?

Суперблок содержит информацию, необходимую для монтирования и управления работой файловой системы в целом (например, для размещения новых файлов).

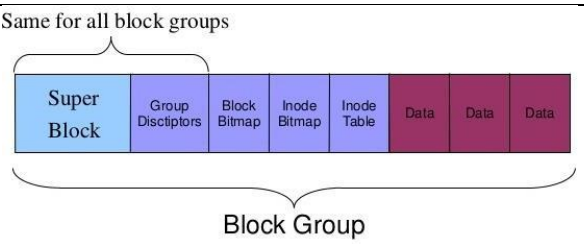
Суперблок — это блок в котором хранятся метаданные файловой системы.

Если вдруг суперблок поврежден, то невозможно будет примонтировать файловую систему. Обычно при загрузке система проверяет суперблок и при необходимости исправляет его, что в результате приводит к корректному монтированию файловых систем.

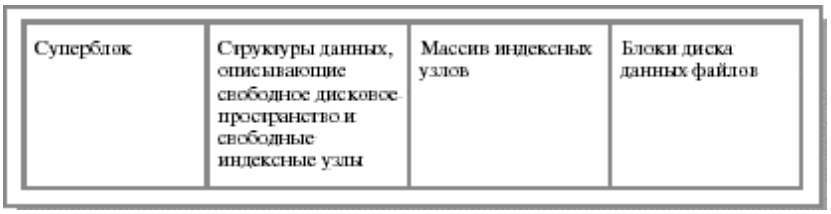
В суперблоке находятся:

1. Счетчик числа свободных блоков.
   Переменная s_nfree.
2. Список свободных блоков.
   Массив s_free[].
3. Счетчик числа описателей файлов.
   Переменная s_ninode.
4. Список описателей файлов.
   Массив s_inode[].

! Ядро Linux работает с размером блока ФС, а не с размером сектора диска (обычно 512 байт).

Основная копия суперблока хранится в самой первой группе блоков. Она названа основной, потому что считывается системой в процессе монтирования файловой системы. !В каждой блоковой группе есть копии суперблока.	

Структура ФС на диске



8. Что представляет собой список свободных блоков?

Дисковое пространство, не выделенное ни одному файлу, тоже должно быть управляемым.

Когда процесс записывает данные в файл, ядро выделяет из ФС дисковые блоки под информационные блоки файла в прямой или косвенной адресации, связывая их с описателем файла в таблице индексных дескрипторов. Суперблок ФС содержит массив s_free, используя его для хранения номеров свободных дисковых блоков.

Программа ядра mkfs организует хранение не используемых в данный момент дисковых блоков в виде списковой структуры, первым элементом которой является массив s_free. Каждый элемент списка хранит номера свободных дисковых блоков аналогично s_free, причём самый первый номер в каждом элементе является ссылкой на следующий элемент списковой структуры.

Mkfs упорядочивает номера свободных дисковых блоков в списковой структуре с целью ускорения поиска блоков. Впоследствии, в связи с непредсказуемостью выделения/высвобождения блоков, mkfs списковых структур не перестраивает.

Первый вариант: часто список свободных блоков диска реализован в виде битового вектора (bit map или bit vector). Каждый блок представлен одним битом, принимающим значение 0 или 1, в зависимости от того, занят он или свободен. Например, 00111100111100011000001 ... .

Главное преимущество этого подхода состоит в том, что он относительно прост и эффективен при нахождении первого свободного блока или n последовательных блоков на диске.

Другой не очень эффективный вариант: связать в список все свободные блоки, размещая указатель на первый свободный блок в специально отведенном месте диска, попутно кэшируя в памяти эту информацию.

Еще метод: свободное пространство можно рассматривать как файл и ввести для него соответствующий индексный узел.

9. Что представляет собой список свободных описателей файлов?

Структура индексного дескриптора: вся информация о файлах, кроме их содержимого и имени, находится в описателях файлов.

Каждому файлу соответствует один описатель. Описатель имеет фиксированный формат и располагается непрерывным массивом, начиная со второго блока. Общее число описателей (максимальное число файлов) задаётся в момент создания ФС.

10. Как производится выделение и освобождение блоков под файл?

Выделение свободных блоков для размещения файлов производится с конца списка супреблока. Когда в списке остается единственный элемент, ядро интерпретирует его как указатель на блок, содержащий продолжение списка. В этом случае содержимое этого блока считывается в суперблок и блок становится свободным. Такой подход позволяет использовать дисковое пространство под списки, пропорциональное свободному месту в файловой системе. Другими словами, когда свободного места практически не остается, список адресов свободных блоков целиком помещается в суперблоке.

Поскольку число свободных inode и блоков хранения данных может быть значительным, хранение двух последних списков целиком в суперблоке непрактично. Например, для индексных дескрипторов хранится только часть списка. Когда число свободных inode в этом списке приближается к 0, ядро просматривает ilist и вновь формирует список свободных inode. Для этого ядро анализирует поле di_mode индексного дескриптора, которое равно 0 у свободных inode.

Если массив s_inode[] в суперблоке не пуст, ядро выделяет очередной описатель файла и назначает его некоторому файлу.

Если массив пуст, ядро просматривает таблицу описателей файлов, выбирая из неё номера свободных описателей файлов, и заполняет ими массив s_inode[], причём запоминается та точка таблицы описателей файлов, на которой закончен просмотр с тем, чтобы при следующем заполнении массива s_inode[] начать с этой точки.

Алгоритм alloc:

1. если суперблок заблокирован, приостановиться до тех пор пока не будет снята блокировка
2. удалить блок из списка свободных блоков суперблока
3. если из списка удалён последний блок:
   - заблокировать суперблок
   - прочитать блок, только что взятый из списка свободных блоков
   - скопировать номера блоков, хранимых в этом блоке, в массив s_free суперблока
   - снять блокировку суперблока
4. переписать в буфер блок, удалённый из массива s_free
5. уменьшить общее число свободных блоков
6. пометить суперблок, как изменённый
7. возвратить буфер, содержащий выделенный блок.

Освобождение блоков:

1. If(список в суперблоке не полон) номер освобождаемого блока включается в список s_free
2. if(список полон) освобождённый блок включается в списковую структуру и ядро записывает в этот блок содержимое массива s_free
3. Массив s_free очищается и его елинственным элементом становится номер этого блока.

11. Каким образом осуществляется монтирование дисковых устройств?

Под файлом в Linux понимается не только поименованная совокупность информации на ВЗУ, но и любое устройство, которое может хранить, поставлять или потреблять информацию. При этом устройство подключается (монтируется) к существующему дереву файловой системы в указанной пользователем точке с помощью команды **mount**, после чего пользователь может обращаться к любым доступным файлам, при этом в имени никак не отражается имя устройства, на котором файл находится или создается.

Пример:

Mount <устройство монтирования> <точка монтирования>
mount –t vfat /dev/fd0 /tc – монтирует файловую систему из раздела fd0 в каталог /tc

Для выполнения монтирования требуются права суперпользователя (sudo – запуск от имени суперпользователя)

Подробнее:
Mount <устройство монтирования> <точка монтирования>
Для выполнения монтирования требуются права суперпользователя.

1. Если пользователь не root, возвратить ошибку.
2. Получить описатель файла для блочного специального файла.
3. Проверить допустимость значений параметров.
4. Получить описатель файла для каталога, где производится монтирование.
5. Если описатель файла не принадлежит каталогу или счётчик ссылок больше единицы:

- освободить описатель файлов;
- возвратить ошибку.

6. Найти свободное место в таблице монтирования.
7. Запустить процедуру открытия блочного устройства для данного драйвера.
8. Получить свободный буфер из буферного кэша.
9. Считать суперблок в свободный буфер.
10. Проинициализировать поля суперблока.
11. Получить корневой описатель файла монтируемой системы, сохранив его в таблице монтирования.
12. Сделать пометку, что описатель файла каталога, являющегося вторым аргументом является точкой монтирования.
13. Снять блокировку с описателя файла каталога, который является точкой монтирования.

12. Каково назначение элементов структуры stat?

Для получения информации о типе файла необходимо воспользоваться системными вызовами stat() (fstat()). Формат системных вызовов stat() (fstat()):

```
#include <sys/types.h>
#include <sys/stat.h>
int stat(const char *name, struct stat *stbuf);
int fstat(int fd, struct stat *stbuf);
```

Оба системных вызова помещают информацию о файле в структурную переменную, на которую указывает stbuf. (в первом случае специфицированным именем name, а во втором – дескриптором файла fd)
Вызывающая функция должна позаботиться о резервировании места для возвращаемой информации; в случае успеха возвращается 0, в противном случае – -1 и код ошибки в errno. Описание структуры stat содержится в файле <sys/stat.h>.

Дополнительно:
Структура stat содержит информацию о файле. Содержится в <sys/stat.h>. Возвращается функцией stat(char *name, stat *info). Описание:

```
struct stat
{
dev_t st_dev;    /* устройство, содержащее файл */
ino_t st_ino;    /* индекс */
ushort st_mode;  /* биты режима */
short st_nlink;  /* число связей файла */
ushort st_uid;   /* пользовательский ID */
ushort st_gid;   /* ID группы */
```

```
dev_t st_rdev;   /* для спец. файлов */
off_t st_size;   /* размер файла */
time_t st_atime; /* время последнего чтения */
time_t st_mtime; /* время последнего редактирования */
time_t st_ctime; /* время последнего изменения статуса */
}
```

Поле st_mode содержит флаги, описывающие файл. Флаги несут следующую информацию:

```
S_IFMT 0170000   - тип файла
S_IFDIR 0040000  - каталог
S_IFCHR 0020000  - байт-ориентированный специальный файл
S_IFBLK 0060000  - блок-ориентированный специальный файл
S_IFREG 0100000  - обычный файл
S_IFFIFO 0010000 - дисциплина FIFO
S_ISUID 04000    - идентификатор владельца
S_ISGID 02000    - идентификатор группы
S_ISVTX 01000    - сохранить свопируемый текст
S_ISREAD 00400   - владельцу разрешено чтение
S_IWRITE 00200   - владельцу разрешена запись
S_IEXEC 00100    - владельцу разрешено выполнение.
```