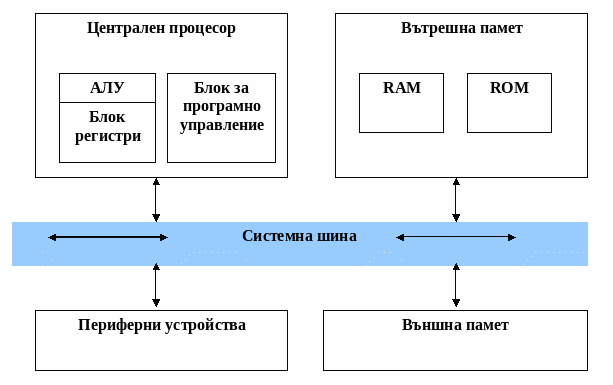
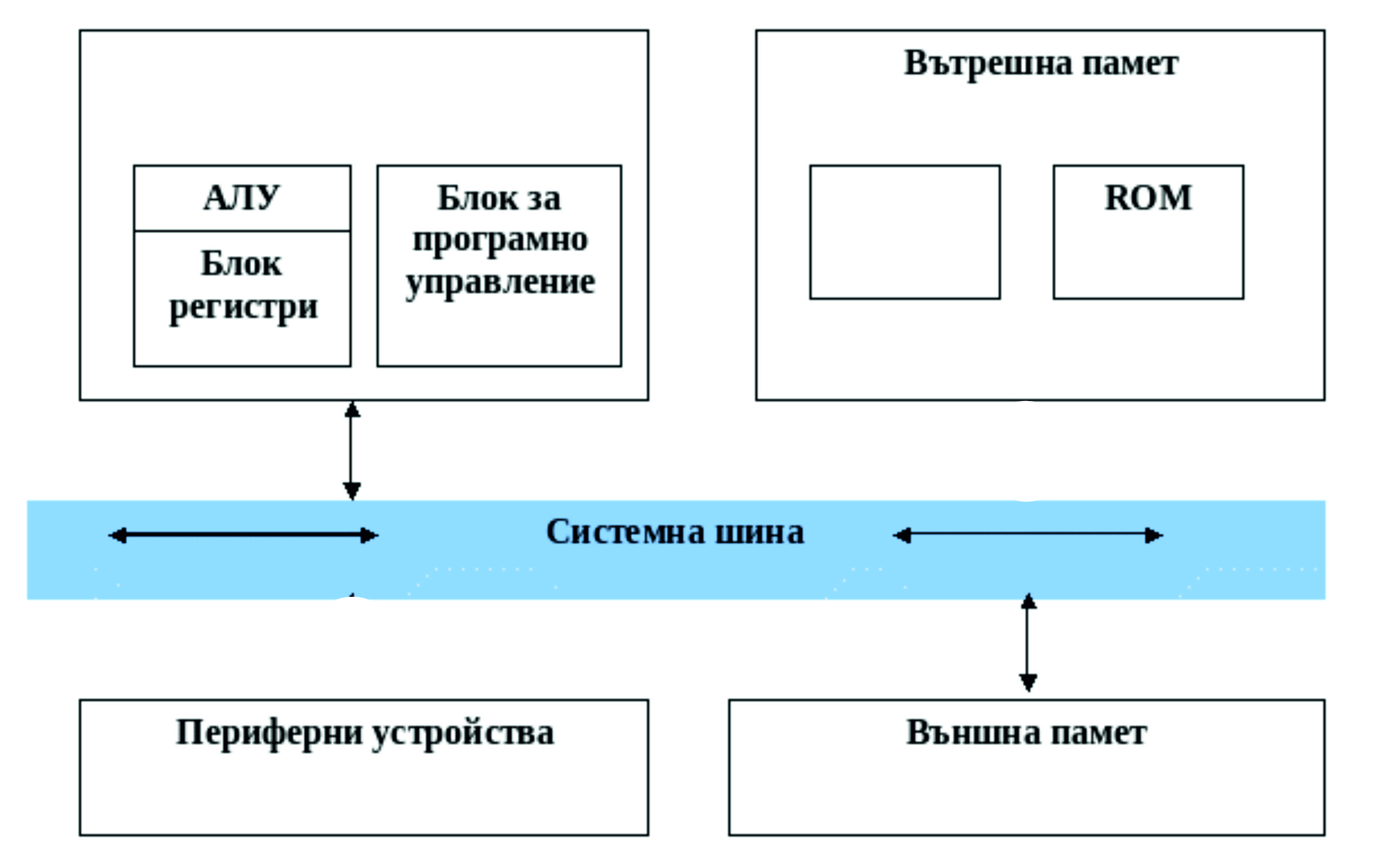
*Изпитна тема № 8:* **Операционни системи**

I. Общата схема на изчислителните машини **Фон- Нойманова архитектура.**

**Допълнете липсващите връзки и наименования на елементи:**





Според нея основни компоненти на компютъра са: Централен Процесор (УУ, АЛУ, регистри), Оперативна памет (RОМ, RAM), шина и Периферни устройства (входни, изходни, запаметяващи, комуникационни, мултимедийни). Стрелките в схемата показват потока на данните.

Фон- Нойман залага основни принципи, които са валидни и за съвременния компютър:

* електронно устройство;
* информацията се представя в двоична бройни система;
* компютърът е програмно управляемо устройство, на базата на алгоритми;

Кратко описание на тези компоненти:

**2. Процесор**– основни характеристики

2.1. тактовата му честота, която се измерва в Mhz или Ghz

2.2. разрядност (данни за 1 такт)- 8,16; 32 и 64

Основните елементи на процесора са: Аритметико логическо устройство, Управляващо устройство, Регистри.

**3. Вътрешна памет:**

3.1.**ROM**(Read Only Memory)- постоянна и неизменяема памет, енергонезависима;

3.2. **RAM**(Random Access Memory)- памет с произволен достъп, временна до изключване на компютъра (енергозависима), измерва се в МВ(128, 256,512) или GB.

**4. Периферните устройства**са:

4.1. входни (клавиатура, мишка, скенер…),

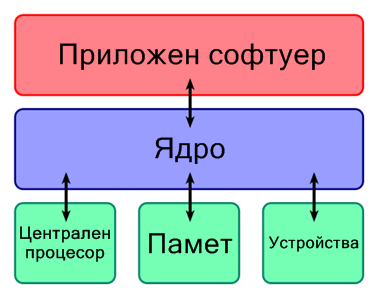
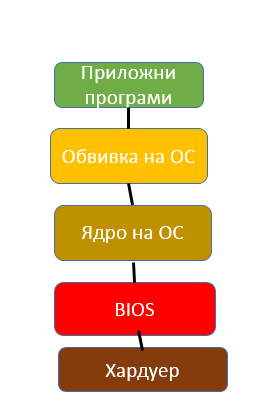
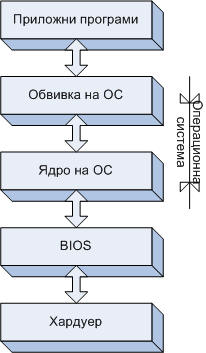
4.2. изходни (монитор, принтер, тонколони…),

4.3. запаметяващи (магнитни(Floppy, Hard disc), оптични (CD, DVD), Flash- памет),

4.4. комуникационни (Lan- карта, модем…),

4.5. мултимедийни (звукова карта, видеокарта).

Всяко едно периферно устройство може да бъде класифицирано по видове.

**Операционна система**

* Дори и най-мощната компютърна конфигурация е безполезна, докато на нея не са инсталирани ОС и програми
* След подбора на хардуерните компоненти и свързването им, първо се инсталира операционната система
* ОС се грижи за всички хардуерни и софтуерни компоненти и взаимодействието им.

**II. Главните програмни части на една ОС са:**

* **Ядро (Kernel)** – зарежда се в оперативната памет при стартиране, веднага след зареждане на BIOS. Ядрото е отговорно за управлението на централния процесор, разпределя паметта.
* **Обвивка (Shell)** – видимия потребителски интерфейс, включващ командите за достъп до ОС.

**Задача**: Проверете какви видове потребителски интерфейс има.

Кои са основите компоненти на графичния потребителски интерфейс?

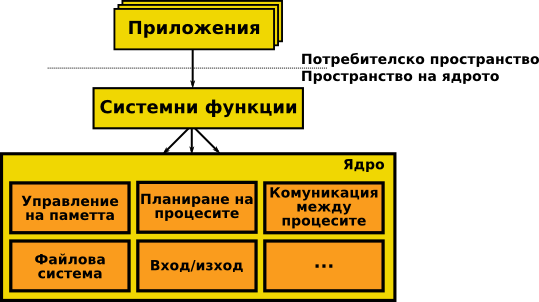
ОС използва и друг вид системен софтуер, който не е част от нея – **драйверите**.

Драйверите указват на ОС как да работи даден хардуер.

Разработчиците на ОС подготвят и възможности за **обновяване (Update**) с цел подобряване на работата.

Когато актуалностите станат много, те се обединяват в общ **пакет за обновяване (Service Pack).**

**Монолитни ядра** - Типична организация на ранните операционни системи. Всяка компонента на операционната система се съдържа в ядрото, тя може директно да комуникира с всяка друга компонента. Висока производителност – няма забавяне за комуникация между компонентите на операционната система. Много трудно се проследяват бъгове. Грешки в една компонента могат да доведат до нежелани ефекти в друга компонента.



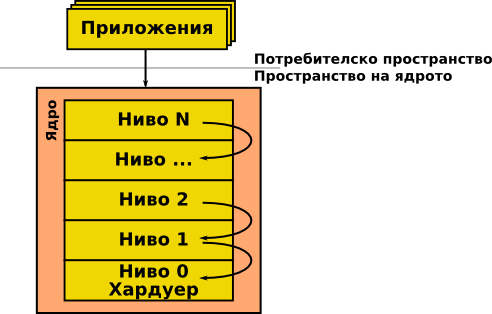
**Многослойна архитектура** - Опит за подобряване на монолитната архитектура.

Групира компоненти с подобна функционалност в слоеве.

Операционната система е разделена на определен брой от слоеве, като всеки слой се изгражда върху основата на по-долните слоеве.

Най-долното (най-ниското) ниво (ниво 0) е хардуера. Всеки слой е организиран така, че да използва функции и услуги само от по-долните нива.

Даден слой може да комуникира само със съседните слоеве. Заявка за изпълнение на някаква операция преминава последователно през слоевете, докато достигне до слоя, който може да я изпълни. Защита между слоевете. 15.02.2022 11клас

Намалява бързодействието на ядрото заради нуждата от организиране на комуникация между слоевете. Най-горното ниво (ниво N) се състои от потребителския интерфейс и потребителските програми.

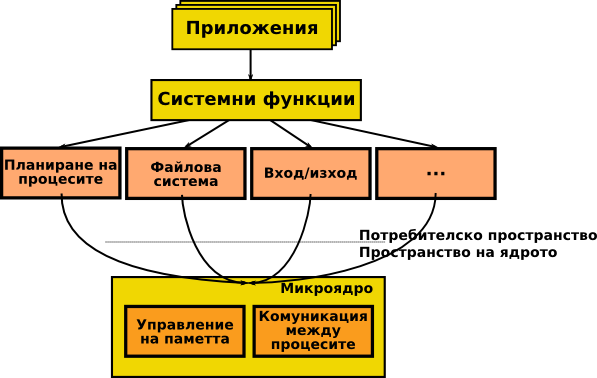
**Микроядра -** Опит ядрото да се направи малко.

Ядрото предоставя само малък брой услуги – управление на паметта и комуникация между процесите. Операционната система става лесно преносима и мащабируема.

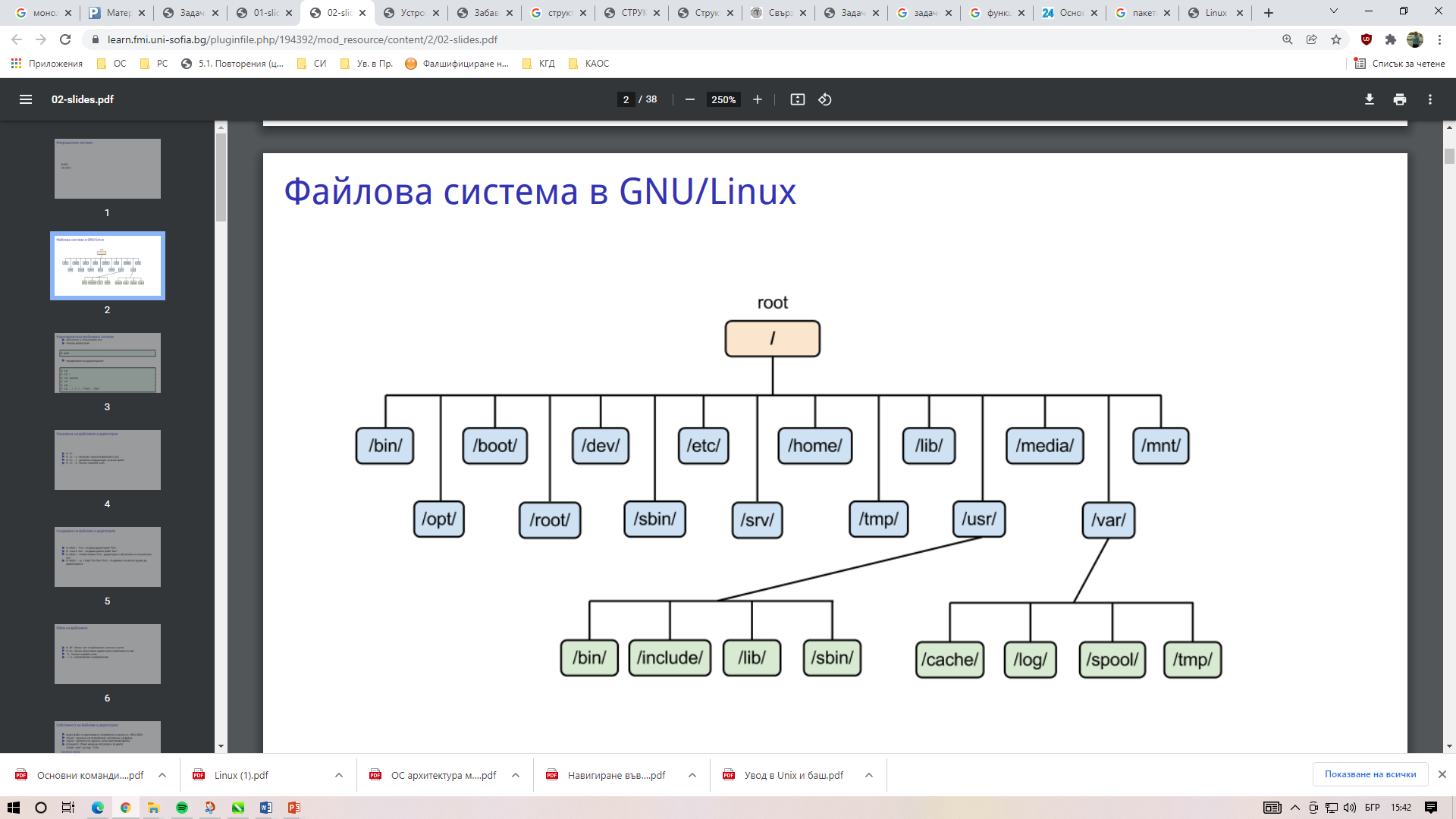
Нараства количеството на комуникацията между процесите, което намалява производителността на ядрото. Максимално количество от услуги, които типично са услуги предоставяни от ядрото, се преместват в потребителското пространство.

Операционната система става по-надежда тъй като по-малко код работи в незащитен режим.

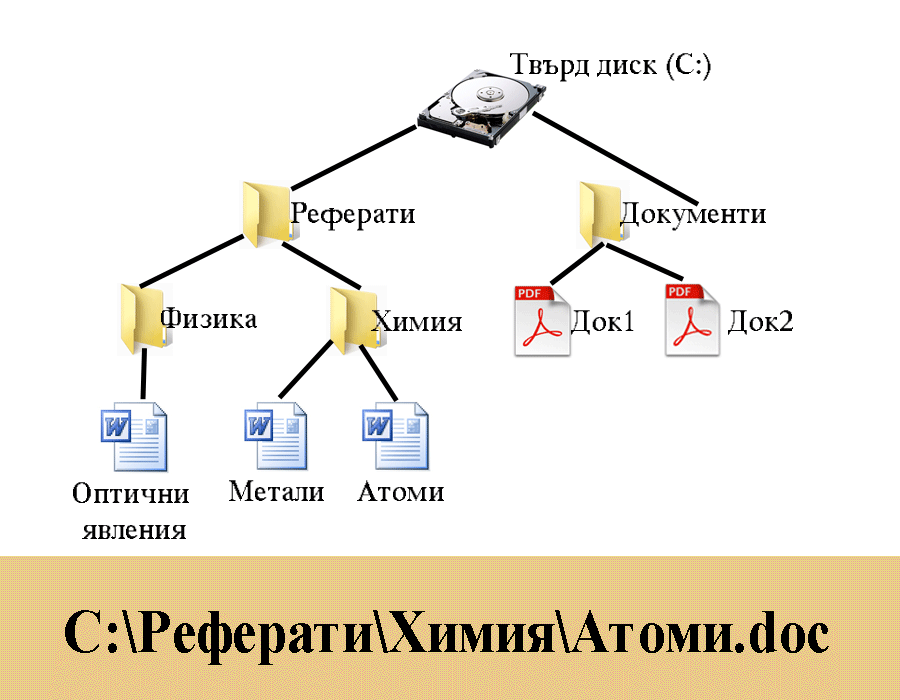


****

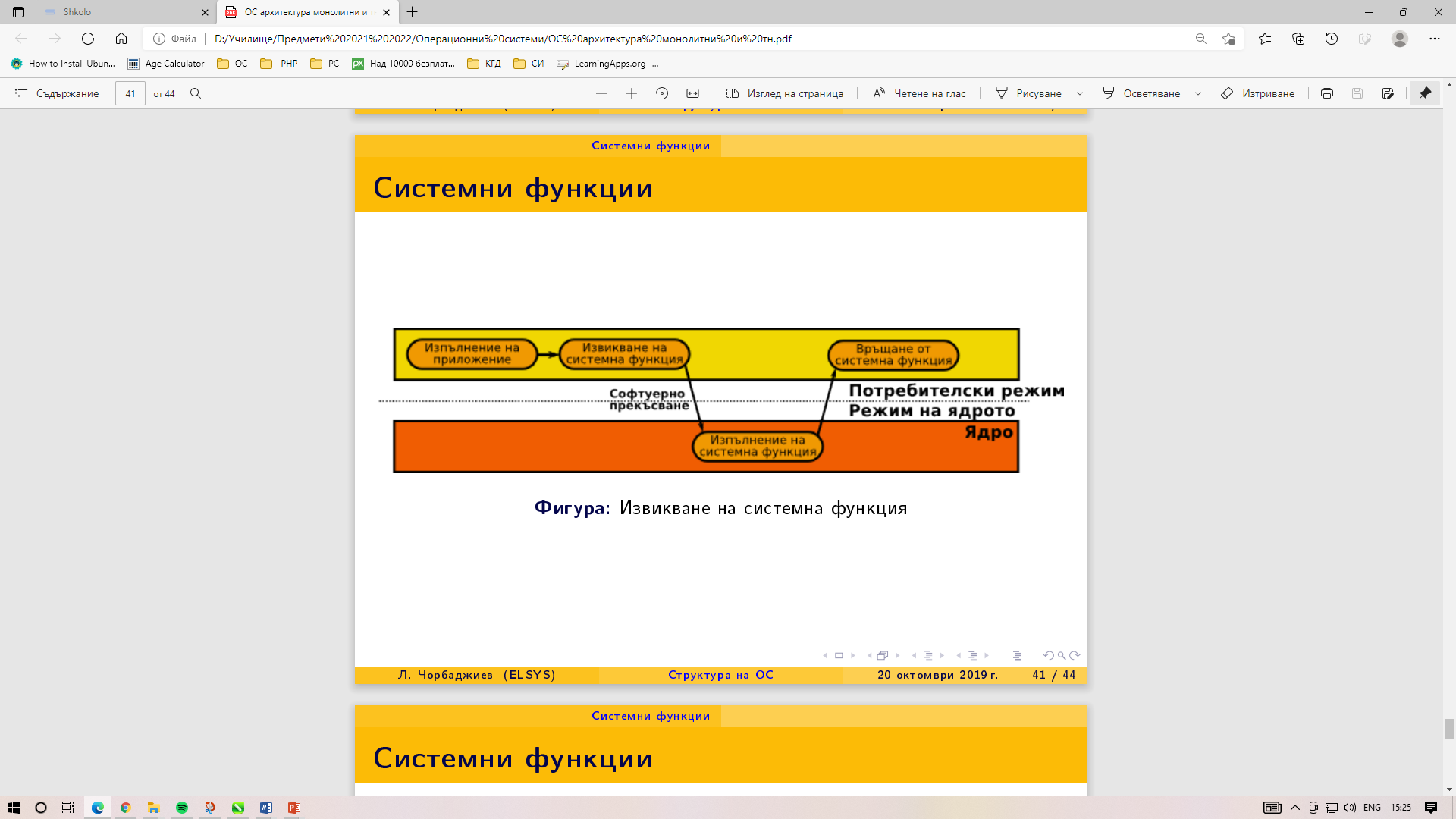
III. Опишете файловата структура на ОС Linux и Windows. Намерете прилики и



Файлова система на Windows



IV. Работата на операционната система се управлява от прекъсванията. Софтуерните грешки и заявките към операционната система генерират софтуерни прекъсвания.



**V.** Всяка Линукс дистрибуция се разпространява с множество по-малки програми и приложения. Всички те са организирани в пакети. Пакета е набор от файлове, които са архивирани и имат определена структура на архива. Тази структура е различна за различните видове пакетни системи.

VI. Процесът е програма, която се изпълнява. Единица работа в рамките на операционната система. Програмата е пасивен обект, процесът е активен. Може да има няколко процеса, които изпълняват една и съща програма. 11 клас на 22.02.2022 г.

Операционната система изпълнява следните дейности, свързани с управление на процесите:

* Създаване (create) и унищожаване (delete) на процеси.
* Спиране (suspend) и възстановяване (resume) на процесите.
* Предоставяне на механизми за синхронизация и комуникация между процесите.

**Виртуална памет**

Linux поддържа виртуална памет, тоест използва диск като разширение на RAM, така че ефективният размер на използваема памет съответно расте. Ядрото ще запише съдържанието на неизползван към момента блок памет на твърдия диск, така че паметта да може да се използва за друга цел. Когато първоначалното съдържание е необходимо отново, те се четат обратно в паметта. Всичко това е направено напълно прозрачно за потребителя; програмите, работещи под Linux, виждат само по-голямото количество налична памет и не забелязват, че части от тях се намират на диска от време на време. Разбира се, четенето и записването на твърдия диск е по-бавно (от порядъка на хиляда пъти по-бавно), отколкото използването на реална памет, така че програмите не стартират толкова бързо. Частта от твърдия диск, която се използва като виртуална памет, се нарича swap space. Много ОС могат да „заблудят“ програмите да използват разпръснати области по твърдия диск или RAM все едно са една непрекъсната област в паметта, наречена виртуална памет.

VII. Услуги в ОС

Някои мрежови услуги като криптирания протокол SSH дори позволяват на участниците в мрежата директен отдалечен до командния интерфейс на компютъра.

Мрежовата архитектура клиент-сървър дава възможност на един компютър (клиент) да се свърже с друг (сървър). Сървърите предлагат разнообразни услуги чрез своите портове или други точки за достъп извън IP адреса на сървъра.

Как да стартираме дадена услуга, която обикновено няма графичен интерфейс. В примерите по-долу, “unit” замества името на процеса или услугата, която искате да стартирате, рестартирате или да настроите да се стартира автоматично при boot.

top - Преглеждайте данни в реално време за процеси, изпълнявани в системата.

pc - Вземете списък с процеси, изпълнявани в системата.

kill - Прекратете изпълнението на процес.

jobs - Показва списък с текущи задачи, изпълнявани във фонов режим.

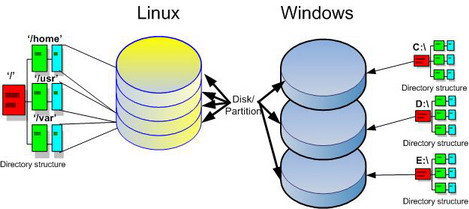
fg - Преместете фонов процес на преден план.

ctrl + z Поставете на пауза текущия процес на преден план и го преместете на заден план.

Control Имаме доста голям контрол върху изпълнението на нашите програми.

*Първо, стартирайте няколко програми на вашия работен плот. След това използвайте ps, за да идентифицирате техния PID и ги затворете.*

VIII. Файловите системи на различните операционни системи.



**IX. Oсновни оператори в shell програмирането**

Един от такъв мощен оператор е "&&" (двойна амперсанда), която е тип верижна команда и когато се използва, изпълнява втората вградена команда при успешното изпълнение на първата команда.

Пример: “apt-get update && apt-get upgrade”, след това успешно изпълнение на командата за актуализиране, командата Upgrade се изпълнява.

Нека се потопим в света на неограничените Linux оператори. Като начало, Linux, като всеки език за програмиране, има основен набор от оператори.

Следва основната класификация или оператори:

Аритметични оператори: използва се за изпълнение на математически операции.

Релационни оператори: просто определя връзката между два операнда.

Булови оператори: известни също като логически оператори, те се използват за извършване на логически операции.

Битови оператори: "&, |, <>", използвани за прости битови операции.

Оператори за тестване на файлове: играе със свойства на файл.

Списък на Linux оператори

Това са няколко от категориите за оператори. Освен това Linux предлага широка гама от команди. Тези оператори основно се изпълняват на Борн Shell. Bourne Shell е оригиналната програма за изпълнение на командите на UNIX, разработена от AT&T, от Стивън Борн, и по този начин името. Сега, нека разберем подробно всеки от тези оператори с примери. За допълнителни примери ще приемем две променливи X и Y, със стойности като 5 и 10.

1. Аритметични оператори

По принцип тези оператори се използват в прости математически изчисления като събиране и умножение.

Имаме 7 аритметични оператора, както следва:

Добавяне (+): Просто добавя стойности на двете дадени променливи. Ex. X + Y ще доведе до 15.

Изваждане (-): изважда стойността на една променлива с другата. Ex. Y - X ще доведе до 5.

Деление (/): Разделя двете променливи. Ex. X / Y ще ни даде 0, 5.

Умножение (\*): Умножава двете променливи. Ex. X \* Y ще доведе до 50.

Увеличение Оператор (++): Просто добавя един към стойността. Операторът за увеличаване работи по два начина, като префикс и като постфикс и въз основа на позицията на оператора, резултатите могат да варират.

За префикс: с оператора преди променлива, Y ++, той ще върне стойността преди увеличаване, напр. Y ++ = 5

За Postfix: с променлива преди оператор, ++ Y, тя ще върне увеличената стойност, напр. ++ Y = 6.

Оператор за намаляване (-): Подобно на увеличението, с изключение на това изтрива една стойност. Работи по два начина.

За префикс: Първо дава стойността на Y, след това извършва операцията decrement, напр. Y - ще доведе до 5.

За Postfix: първоначално намалява стойността с една, след което дава резултата, Ex. –Y ще върне 4.

По-горе са широко използвани аритметичните оператори и сега нека започнем с втория списък от оператори, т.е. оператори за връзка.

2. Релационни оператори

Просто връща или "вярно или невярно", в зависимост от връзката между променливите и се поддържа от Bourne Shell.

По-долу има общо 6 вида релационни оператори:

Равен на (==): Сравнява двете променливи и връща вярно, ако са равни и неверни, ако е друго. Ex. X == Y ще доведе до невярно.

Не е равно на (! =): Подобно на равно на, с изключение на това, че връща вярно, ако стойностите не са еднакви и неверни, ако е друго. Ex. X! = Y ще върне вярно.

По-малко от (<): ако стойността на лявата страна е по-малка от стойността на дясната, тя връща вярно, друго неверно. Ex. X <Y ще върне вярно.

По-голям от (>): връща true, ако стойността на лявата страна е по-голяма от дясната стойност, в противен случай false. Ex. X> Y ще върне невярно.

По-малко или равно на (<=): вярно, ако лявата стойност е по-малка или равна на дясната стойност.

По-голямо или равно на (> =): вярно, ако лявата стойност е по-голяма или равна на дясната стойност.

Сега, след като разбрахме релационните и аритметичните оператори, нека проучим булевите и битовите оператори.

**3. Булeви оператори**

Напълно поддържани в Shell, булевите оператори се използват в комбинация за по-добри резултати от търсенето. И, ИЛИ и НЕ са булеви оператори и използването на резултатите от тези оператори спестява време.

И: да стесните търсенето. Ex. Тестови оценки И окончателни оценки.

ИЛИ: разширете търсенето. Ex. Безплатни курсове ИЛИ по Rs.499.

НЕ: за премахване на ключови думи. Ex. Последни допълнения НЕ над Rs.499.

4. Битови оператори

Донякъде подобно на логическите оператори, с изключение на битовите оператори, работещи на по-ниското ниво при Binary Представяне на данни. Тук и двете променливи (дясна и лява стойност) трябва да са Цели.

AND (&): сравнение между два бита. Връща 1, ако и двата бита са 1, в противен случай 0.

ИЛИ (|): сравнява два бита и връща 1, ако битовете са допълващи се, 0.

**Х. Виртуализация и контейнери**

Виртуалните машини са част от средите за виртуализация. Контейнерът, от друга страна, е специфичен вид стратегия за виртуализация, която предлага своите предимства и недостатъци в различни видове платформи за виртуализация, включително хоствани мрежи и облачни изчислителни системи.

Файл

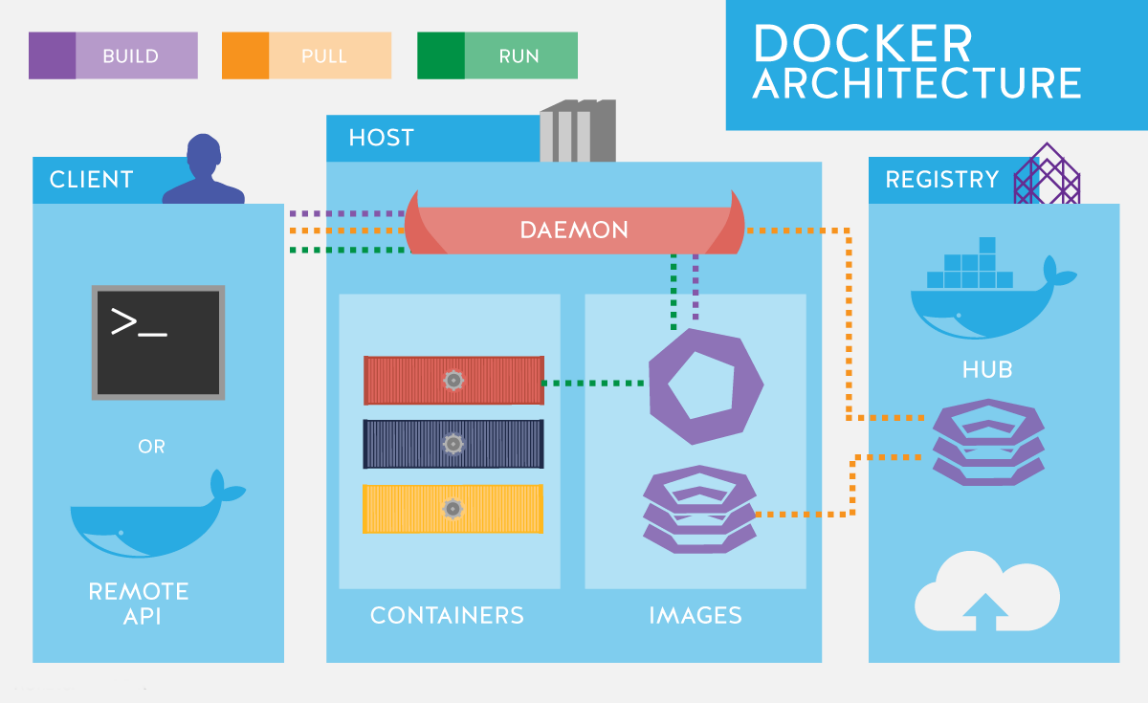
Виртуализация

Файл

С виртуализацията на контейнерите виртуализацията се извършва на ниво операционна система, а не на ниво хардуер. Отделни случаи споделят части от ядрото, така че данните получават маршрутизация по различен начин.

С виртуализиращата технология, пакетът, който може да бъде прехвърлян е виртуална машина и включва цялостна OS както и приложението. Физически сървър поддържащ три виртуални машини ще има компютърен софтуер или хардуер, който създава и

стартира Виртуални машинии три отделни операционни системи, работещи на него.

**Виртуална машина, Това е емулация на компютърна система. Освен това, VM се основава на компютърна архитектура и осигурява функционалността на физическия компютър. Тя включва специализиран хардуер, софтуер или комбинация. За създаване и стартиране на виртуални машини се използва хипервизор или монитор на виртуална машина.** 

**А какво точно е контейнер?**

Контейнерът е стандартна единица софтуер, който събира код и всички негови зависимости, така че приложението работи бързо и надеждно от една компютърна среда в друга. Изображението на контейнера в Docker е лек, самостоятелен, изпълним пакет от софтуер, който включва всичко необходимо за стартиране на приложение: код, време на изпълнение, системни инструменти, системни библиотеки и настройки.

Предлага се както за Linux, така и за Windows-базирани приложения, като софтуерът в контейнера винаги ще работи еднакво, независимо от инфраструктурата. Контейнерите изолират софтуера от неговата среда и гарантират, че той ще **работи равномерно, въпреки разликите между разработката и етапа на тестване например.**

|  |  |
| --- | --- |
| **Критерии за оценяване на изпитна тема № 8** | **Максимален брой точки** |
| 1. Посочва, различава и демонстрира знания за отделните хардуерни компоненти на компютърна система. | 12 |
| 2. Обяснява структурата на операционната система. Демонстрира знания за архитектурата на операционните системи – монолитни, многослойни и архитектура с микроядра. | 10 |
| 3. Прави заключения и изводи за файловата структура на ОС при конкретна поставена задача. | 8 |
| 4. Посочва, обяснява и демонстрира команди, чрез които се показва функционалността на операционните системи. | 12 |
| 5. Обяснява пакетните системи в ОС. | 4 |
| 6. Диференцира процесите в операционните системи. Демонстрира знания за виртуалната памет на конкретна операционна система. | 14 |
| 7. Изброява и обяснява услуги в ОС. При конкретна поставена задача за стартиране и спиране на услуга ръчно или по график, избира правилния начин. | 14 |
| 8. Прави заключения и изводи за файловите системи на различните операционни системи. | 8 |
| 9. Обяснява основни оператори в shell програмирането. Демонстрира знания за създаване на shell скриптове. | 10 |
| 10. Прави изводи за виртуализация и контейнери. | 8 |