4) Co rozumíte pod pojmem **Access Control List, ACL**? Kde se používá a jaké má vlastnosti?

Ku kazdemu suboru je pripojeny zoznam pristupovych opravneni -> usporiadane dvojice (domena,operace)

Plne ACL = viac prav(mazanie, zmena opravneni), negativne zaznamy; lubovolny pocet zaznamov, dynamicka dedicnost -> propagace zmien do podadresarov

Zjednodusena verzia: v UNIXE read/write/execute entity:user/group/other

autorizace je povolení přístupu/akce na základě platného ověření identity (autentizace).

5) Popište, jakým způsobem pracuje **operační systém s periferiemi** a uveďte dva příklady rozdílných periferií.

Periferie = zariadenia, ktoresa pripojuju k pocitacu (konektory, sloty..)

* Vstupne/ vystupne
* Komunikacia s inymi zariadeniami
* Zariadenia na permanentne ukladanie dat

Zpristupnene pomocou prislusneho API -> moznost vymeny. OS ovlada periferie cez ich ovladace, tie maju vseobecne API -> vytvorenie abstrakcie periferie. Ovladace zariadeni vystupuju navonok jednotne prace skrz svoje API.

Napr praca s diskom, grafickou kartou, sietove karty, mys, klavesnica

Zaclenenie ovladaca do jadra moze byt: kooperativne/hierarchicke

Forum : U této otázky jste měli vysvětlit, že OS ovládá periferie skrze jejich ovladače, které mají nějaké obecné API a tedy vytváří jakousi abstrakci periferie. Ovladače různých zařízení pak vystupují navenek (z pohledu OS) jednotně právě skrze své "univerzální" API. Jako příklad dvou rozdílných periferií jsem akceptoval pokud jste uváděli periferie dvou různých kategorií (např. monitor, myš; klávesnice, tiskárna...), nikoliv však periferie stejné kategorie (myš, klávesnice -- obojí vstupní zařízení). Krásnou ilustrací toho jak to v operačních systémech vypadá s periferiemi je třeba standard HID či ALSA.

6) Srovnejte stručně **polling** a přerušení. Uveďte a stručně okomentujte nějaký systém, kde se používá polling.

9) Co je to **přerušení** a jaký má význam? Stručně uveďte princip přerušení.

Prerusenie je mechanizmus ako prerusit proces na zaklade externej priciny

* Prerusenie od casovaca – timeout
* Prerusenie od procesora – delenie nulou
* Prerusenie od periferie – klavesnica, mys...

Pri preruseni je nutne ulozit stav procesu a zapamatat si miesto navratu-obsluha v kerneli -> zaistuje serializaciu a bezpecnost + vyvolava prepnutie kontextu

Po obsluhe sa pokracuje v povodnej praci -> pri viacerych zdrojoch a pricinach je potrebne si zapamatat zdroj a typ

Urovne : nemaskovane, aktualne spracovavane prerusenie, maskovane

Polling je opakovany dotaz na nejaky stav/udalost -> aktivne -> zatazuje procesor viac ako prerusenie

Oplati sa v pripade, ked udalost, na ktoru sa pytame nastane skorej ako by prebehlo obsluzenie prerusenia

Vezměte si např. způsob, jakým funguje web - není to server

facebooku kdo vám zasignalizuje novou zprávu, ani server googlu kdo Vás upozorní na nový email. Váš prohlížeč se periodicky dotazuje, zda není něco nového. To je polling. A existuje i na hardwarové úrovni, např. čip v klávesnici neustále skenuje, zda není stisknuta nějaká klávesa.

10. Charakterizujte **bezdrátové sítě** (jaké mají vlastnosti) a srovnejte se

sítěmi, využívajícími pevné medium (metaliku nebo optiku).

- Siete používajú ako médium na prenos signálu rádiové, mikrovlnné, alebo infračervené vysielanie.

-data su rozdelene na male pakety -> kazdy prechadza sietou samostatne -> mozu prijst v roznom poradi -> zoradenie zabezpeci protokol napr TCP

- neni treba uchovavat stav siete (spojovane udrzuju stav siete) -> vacsia robustnost

- problematickejsia implementacia ako u spojovanych sieti

- u spojovanych sieti je spojenie udrzovane pocas celej doby spojenia -> plytvanie pasmom

11. Stručně charakterizujte **statické směrovací algoritmy** v počítačových sítích.

Jaké jiné typy směrovacích algoritmů znáte? A které se používají v Internetu?

**10.Smerovanie - statické a dynamické**, ine pohľady, čo používame v Internete

-Staticke algo maju staticke tabulky, vhodne pre staticku topologiu, na zaciatku prenosu obsah tabuliek urci spravca

- napr dynamicke -> tabulky sa menia = aktualizovane v reakcii na zmeny v topologii

1. dynamické smerovanie, smerovacie informácie sa učí od iných routrov, pri veľkých sieťach
2. statické smerovanie, sieťový administrátori konfigurujú ručne informácie o smerovaní, malé siete

stochaticke/deterministicke, jednocestne/viaccestne, distribuovane/centralizovane, krok za krokom/ zdrojove – podciarknute internet

12. Stručně diskutujte, jaký mají podle Vás informační technologie dopad na

výrobní a komerční procesy (které změny považujete v této souvislosti za

podstatné, resp. které podle Vás nejvíce ovlivňují výrobu a komerční činnost obecně).

1) Popište stručně základní vlastnosti **peer-to-peer** systémů a uveďte základní rozdíly proti systémům **klient-server**.

P2P je distribuovany heterogenny system uzlov, kt si navzajom poskytuju sluzby -> vzajomne prepojene identicke entity, kt medzi sebou komunikuju. Kazdy peer je klientom a serverom naraz -> poskytuje sluzby ostatnym aj vyuziva sluzby ostatnych. Ma silno dynamicku topologiu, vdaka absencii centralnych uzlov netrpi single-point-of-failure -> robustnejsie. S rastucou kapacitou uzivatelov rastie aj prenosova kapacita. Napr Skype, BitTorrent

Vlastnosti P2P : decentralizovana sprava, samoorganizacia, heterogenita (peerovia mozu byt na roznych platformach), skalovatelnost (neni mozne pretazenie centralnej entity ako u k/s), dynamika, zdielanie zdrojov

P2P oproti k/s su narocnejsie na implementaciu, komplikovanejsie bezpecnostne protokoly

Sprava k/s je prehladnejsia kvoli koncentracii komunikacie v jednom bode

k/s model je limitovany HW parametrami serveru – P2P ma rastucu kapacitu pri pridavani novych uzlov

k/s za bezpecnost zodpoveda server co je jednoduchsie ako pri P2P, kde je zodpovednost rozlozena medzi peery

spolahlivost k/s je zavisla od behu systemu, P2P je do velkej miery redundantny -> 1 funkcionalitu poskytuje viac peerov

11. Popiste **vlastnosti P2P**. V com su lepsie nez K/S.

Viz predchazdajuca otazka

Lepsie su v : lepsia skalovatelnost -> nehrozi pretazenie centralnej entity ako u k/s. S pribudajucimi peermi sa zvysuje kapacita systemu. P2P nie su zavisle od behu 1 entity (servra) ale su do velkej miery redundantne -> 1 funkcionalitu poskytuje viac peerov

2) Co rozumíte pod pojmem **rasterizace písma** a proč jsou s rasterizací problémy?

Jak lze ten problém vyřešit? (stačí jeden příklad, ale je potřeba stručně

vysvětlit princip)

Pri rasterizacii vznikaju chyby hlavne na hranach, stracaju sa detaily, objavuju sa artefakty, nejednoznacne hranice polygonov -> tieto chyby zhorsuju citatelnost pisma, preto sa po rasterizacii aplikuje antialiasing -> text je stale rozmazany -> **hinting** (= predpocitane parametry pre dany font a rozlisenie) – technika interpolacie vybranych pixelov, ktora umoznuje vykreslenie cistejsieho a ostrejsieho obrazu

**Antialiasing** – predfiltrovanie – aplikuje sa behom rasterizacie, odstranuje zubate hrany vyhladenim ostrych prechodov v obraze

* technika eliminacie artefaktov, pixle su zafarbene intenzitou priamo umernou ploche, ktorou su rastrovym geometrickym objektov pokryte

pri vykresleni vektoru do rastru dochadza k rozpadu tvaru -> obtiazne citatelne -> antialiasing -> neziaduce ucinky -> hinting

3) Srovnejte **překladače a interprety** programovacích jazyků. Co rozumíte pod

pojmem **just-in-time** překladače?

JIT je sucast niektorych interpretrov -> optimalizacia -> kod ktory sa bude opakovane paralelne vykonavat sa s vykonavanim kompiluje = dynamicke prekladanie = preklad pocas vykonavania (aby ho nebolo treba opakovane interpretovat, ale aby sa dal rovno pouzit) -> kusok nie cely program -> zrychlenie interpretru

Interpreter je program ktory priamo vykonava zapis ineho programu v zdrojovom kode konkretneho prog jazyka -> nie je nutne vykonavat preklad do strojoveho kodu -> zlepsena prenositelnost medzi platformami -> Python, Basic

Kompilator je program ktory podla pravidiel daneho prog jazyka prelozi algoritmus = postupnost prikazov do medzikodu a ten do strojoveho jazyka alebo priamo don.

7) Jaké problémy činí **výplň ploch na rastrových displejích**? Jak se tyto problémy řeší?

Vypln ploch je jeden z najcastejsich problemov rastroveho obrazu -> treba zafarbit vsetky pixle danej oblasti

- definícia oblasti: - všetky pixle danej farby

- v danej vzdialenosti od pixla

- definovaná polygónom

- definicia susedneho pixelu : 4-smerna(spolocna hrana)

8-smerna(spol vrchol/hrana)

Techniky vyplnovania:

* zaplavove vyplnovanie = zvol 1 pixel vo vnutri oblasti a rekurzivne vyfarbuj susedov
* riadkove = rekurzivne po riadkoch nie pixeloch -> efektivnejsie
* paritne vyplnovanie = najde priesecniky riadku s polygonom, zoradi podla polohy osi x, vyfarbi parne useky

Problemom je nejednoznacnost hranice -> vyplne, lebo mame viac sposobov jako definovat plochu aj susedny pixel

8) Co to je a k čemu slouží **Aplikační programové rozhraní (API)?**

Definuju sposob pristupu k OS a inym sluzbam. Pozostava z definicii funkcii, datovych struktur a tried nejakej kniznice a pravidiel, kt. hovoria ako mozu komunikovat komponenty nejakeho softweru

* ovladac implementuje API => jednotlive ovladace su zamenitelne

ucel: lahka uprava kodu prenositelnost

preklad medzi sluzbami vyssej a nizsej urovne -> prevod typov/struktury parametrov, prevod medzi sposobmi predavania parametrov (&)

definuje napr pracu s pamatov, so suborami

1) Uveďte, co rozumíte pod pojmem **von Neumannova architektura počítačů** a stručně tuto architekturu popište.

„Princentonska architektura“

- pocitac sa deli na zakladne hardwareove jednotky : I/O zariadenia, aritmeticko-logicka jednotka, radic a operacna pamat.

V porovnani s Hardvardskou koncepciou je tu operacna pamat zdielana -> data aj instrukcie su umiestnene v pamati spolocne -> sekvencne spracovanie (u Harvardskej paralelne => von Neumannova zbrzdila vyvoj paralelnych zariadeni). Prerusenie sekvencenho spracovavania pomocou podmienenych a nepodmienenych skokov.

Radic prevadza postupnost instrukcii na postupnost signalov. Udaje aj instrukcie su vyjadrene binarne. Programom riadene spracovanie dat sa robi samocinne. Radic vysiela signaly a dostava stavove hlasenia. Data tecu z ALU do radica = riadiaca jednotka spracovava jednotlive instrukcie ulozene v pamati -> ich vlastne provadeni ma na starosti ALJ.

2) Napište, čím se liší **swapování, stránkování a segmentace paměti**.

Suvisia s virtualizaciou pamate = technika spravy pamate, pri ktorom je umoznene procesorom pridelit pamat, ktora je vacsia ako fyzicka operacna pamat -> nudzove riesenie ked pristup na disk je pomalsi ako pristup do OP

Strankovanie = adresovacia technika operacnej pamate, ktora umoznuje vyuzivanie virtualnej pamate, fyzicka operacna pamat sa rozdeli na useky pevnej dlzky = ramce a virtualna pamat sa rozdeli na rovnake velke useky = stranky. Strankovanie je potom pridelovanie ramcov pamate strankam podla potreby (tie menej pouzivane su ulozene na pomalom mediu na disku a ostatne su vlozene v ramcoch rychlej OP)

Segmentacia = je sposob spravy a ochrany pamate, pamat je rozdelena na segmenty (ma adresu bazu offset) roznej dlzky a kazdy proces moze pristupovat len k segmentu, ktory mu bol prideleny

Swapovanie = vymena stranok medzi fyzickou a virtualnou pamatou (swap out = ulozenie stranky na disk, swap in = nacitanie stranky do pamate)

Z fora: Virtuální paměť (virtualizace paměti) není nějaký soubor na disku, je to způsob adresace paměti programu pomocí logických adres, které jsou mapovány na adresy fyzické. To, spolu se swapováním stránek umožňuje na první pohled nafouknout RAM na více než je dostupná velikost. Dalším důsledkem tohoto je pak fakt, že virtuální adresní prostor je souvislý, kdežto fyzický souvislý být zdaleka nemusí. A více instancí téhož programu může sdílet jednu oblast paměti s programem, ale každý mít vlastní data (kód programu je ve všech jeho instancích mapován do stejných fyzických rámců). Co se velikosti týká, lze polemizovat že správná odpověď se odkáže na velikost adresního prostoru - tj. 2^32 u 32-bit počítačů (4GB) a nebo 2^64 u 64-bit počítačů. Akceptováno ale bylo jakékoliv rozumně zdůvodněné číslo (při příliš velkém požadavku dochází k tzv. overcommitu - tj. více požadavků na paměť, než má systém šanci rozumně uspravovat) - zpravidla se vyskytovala odpověď 1,5 násobek velikosti RAM.

3) Srovnejte **monolitickou a modulární architekturu operačního** systému. Má

monolitický systém nějakou výhodu ve srovnání s modulárním?

Monolit = povodne Os, nejasne rozlisenie funkcii, zle rozsiritelne/udrziavatelne, 1 velky proces bez strukturacie, len funkcie a procedury, vsetok kod bezi v rovnakom jadrovom priestore = kerneli

Výhoda monolitu je právě vyšší teoretický výkon, to že je ušitý na míru

konkrétnímu systému

Modularny = rozdeleny do modulov(zapuzdrovanie – encapsulacia funkcii), ktore medzi sebou komunikuju (moze byt komplikovana -> zvysovanie vykonu), mozu sa bez obmedzenia pridavat – riziko vzniku fatwaru. Lepsia udrzba – vymena po jednom module

10. **smerovanie / prepinanie**

Smerovac spaja rozne siete -> pracuje na sietovej vrstve ma v sebe smerovavie tabulky s IP adresami podsieti, ktore spravuje. Podla IP adresy smeruje pakety z jednej podsiete do druhej. Ma svoju vlastnu IP napr router

Prepinac – pracuje na nizsej vrstve(linkovej = vrstva datoveho spoja) je urceny pre lokalne siete na zdielanom mediu napr u ethernetu. Transparentne pre IP – prepinac nema svoju IP adresu napr switch = sluzi ako centralny prvok v hviezdicovej topologii. Udrziava si tabulku MAC adries zariadeni pripojenych k portom. Pomocou MAC adresy rozpoznava kam maju byt pakety dorucene -> nepreposiela ich na vsetky porty sucasne -> znizenie prebytocnych dat v sieti

12. Otazka na zamyslenie "Veda a IT"

7) Co je to **světlo**? Co **jsou barvené prostory**? Proč se jimi v grafice zabýváme.

Svetlo je elektormagneticke vlnenie charakterizovane vlnovou dlzkou a intenzitou – viditelne spektrum 400nm(fialova) - 700nm(cervena)

Farebne priestory:

* farba je schopnost povrchu odrazat alebo pohlcovat svetlo
* farebne priestory = mnozina farieb pevne napamapovana na farby realneho sveta => referencne farby + funkcia na miesanie
* gamut = cast farebneho priestoru, kt je nieco/niekto schopne vnimat, reprodukovat – mnozina zobrazitlenych farieb => vysek farebneho priestoru

Sluzia nam na co najvernejsie vykreslenie realneho sveta -> farby, tiene.

Farebne priestory potrebujeme aby sme mohli standardizovat zariadenie a aby sa mohli explicitne odkazovat na konkretne farby

8) Co rozumíte pod pojmem **programovací jazyky**? Jakou roli hraje překlad a do čeho se překládá?

Je to formalny jazyk, ktory sa pouziva k vyjadreniu nejakej abstraktnej myslienky. Ma svoju syntax a semantiku. Preklad sa pouziva na prevedenie do ineho jazyka alebo do strojovo citatelneho formatu => strojoveho kodu.

Je to nastroj na systematicky opis postupu, vypoctu, spravania, zmeny stavu, opis objektov ich vlastnosti a struktur. Sluzia na opis problemu a vykonanie pomocou programovatelneho technickeho zariadenia

10) Při diskusi **programovacích jazyků** byla zmíněna míra **abstrakce**. Vyjmenujte základní stupně abstrakce v kontextu programovacích jazyků a velmi stručně uveďte, co je pro jednotlivé stupně typické.

Přibližně polovina odpovídala na otázku tak, že vyjmenovala a diskutovala

programovací paradigma (procedurální, objektový, atd.). To ale není míra

abstrakce (strojový kód, assemblery, vyšší jazyky [, bytecode]).

Dále jste zapomínali na podstatný detail jazyků vyšších - agregované datové typy (struktury), či různé typy cyklů a funkcí nebo vlákna.

10.Čo je **port** v počítačových sieťach, načo sa používa a na akej vrstve.

Transportna vrsta

= adresa (cislo) poskytovanej sluzby napr http = port 80

11**. Digitálny podpis** ako sa realizuje

Zaistuje nepopieratelnost povodu, pouziva sa asymetricka kryptografia

Vznikne tak ze A vytvori hash (otisk pevnej dlzky – MD5, SHA1 – jedinecny pre kazdu spravu, nejde z neho rekonstruovat spravu) zasifruje svojim sukromnym klucom -> overenie: zo spravy premnenlivej dlzky vytvori B otisk potom zobere prilozeny hash a desifruje ho verejnym klucom A -> podpis je pravy ak sa novovzniknuty otisk a desifrovany zhoduju. Tymto sa A preukaze B ale nie naopak

12. Zamyslenie: Aká vedná disciplína bola najviac ovplyvnená vďaka masovému

nasadeniu výpočetní techniky

1) Jakými částmi či vlastnostmi je charakterizován **procesor**? Stručně zdůvodněte, komentujte.

Casti: ALU, FPU -> zakladne sekvencne jednotky, registre, cache, radic, MMU, mustky, radice zbernic

* ked obsahuje viacej sekvencnych jednotiek = vnutorny paralelizmus = viacjadrovy procesor

ALJ – pocita aritmeticke a logicke operacie

* od 80 rokov aj koprocesor FPU -> aritmeticke operacie s pohyblivou desatinnou ciarkou

MMU – memory managment unit -> zodpoveda za preklad virtualnej adresy generovanej programom na fyzicke adresy hlavnej pamate

Radic meni instrukcie na postupnost signalov, riadi a komunikuje s ostatnymi castami procesora

Registre predstavuju najrychlejsiu pamat procesora ale aj tu najmensiu -> kratkodobe uchovavanie – adresy, medzivysledky

Cache – vyrovnavacia pamat, mala kapacita, sluzi na zrychlenie toku dat medzi procesorom a operacnou pamatou -> stupne L1, L2, L3

Vlastnosti: instrukcny cyklus(doba na vyber prevedenie instrukcie 1 jednotkou), takt (vnutorne hodiny), velkosti cache, instrukcna architektura, cena, sirka datovej zbernice(8/16/32/64), sirka adresnej zbernice

+ je charakterizovany instrukcnou sadou (RISC,CISC)

Typy : univerzalne, specializovane (vektorove = nepracuju so skalarmi ale s vektormi => su schopne spracovat viac dat naraz napr pri zistovani pocasia, graficke = vhodne pre zlozite vypocty napr na hry alebo aj na vyskum, embedded = jeden ucel napr chladnicky)

2) Co je to **virtuální paměť** a jakou má úlohu v běžné architektuře počítačů (proč byla zavedena)? Jakou má zpravidla velikost?

Forum: Virtuální paměť (virtualizace paměti) není nějaký soubor na disku, je to způsob adresace paměti programu pomocí logických adres, které jsou mapovány na adresy fyzické. To, spolu se swapováním stránek umožňuje na první pohled nafouknout RAM na více než je dostupná velikost. Dalším důsledkem tohoto je pak fakt, že virtuální adresní prostor je souvislý, kdežto fyzický souvislý být zdaleka nemusí. A více instancí téhož programu může sdílet jednu oblast paměti s programem, ale každý mít vlastní data (kód programu je ve všech jeho instancích mapován do stejných fyzických rámců).

Co se velikosti týká, lze polemizovat že správná odpověď se odkáže na velikost adresního prostoru - tj. 2^32 u 32-bit počítačů (4GB) a nebo 2^64 u 64-bit počítačů. Akceptováno ale bylo jakékoliv rozumně zdůvodněné číslo (při příliš velkém požadavku dochází k tzv. overcommitu - tj. více požadavků na paměť, než má systém šanci rozumně uspravovat) - zpravidla se vyskytovala odpověď 1,5 násobek velikosti RAM.

MMU (memory managment unit) je cast procesora zodpovedajuca za preklad virtualnych adries generovanych programom na fyzicke adresy na disku.

3) V souvislosti se způsobem implementace hovoříme o **monolitickém kernelu**

operačního systému. Jaké další způsoby znáte? Jak byste charakterizovali dnes používané kernely (MS Windows, jádro Linuxu, GNU Hurd, MacOSX, ...)

Monolitické jadro je druh [jadra](https://sk.wikipedia.org/wiki/Jadro_(informatika)) [operačného systému](https://sk.wikipedia.org/wiki/Opera%C4%8Dn%C3%BD_syst%C3%A9m), ktorého všetok kód beží v rovnakom (jadrovom) pamäťovom priestore, ktorý sa anglicky označuje ako kernel space. Tým sa líši od takzvaného [mikrojadra](https://sk.wikipedia.org/wiki/Mikrojadro), ktoré väčšinu tradičných činností monolitického jadra, ako je napr. správa súborového systému, implementuje v procesoch, ktoré bežia v užívateľskom pamäťovom priestore (user space).

[Unixové](https://sk.wikipedia.org/wiki/Unix) jadro je typický príklad monolitického jadra a jeho pokračovatelia (FreeBSD, Linux alebo Solaris) nie sú iní. Na rozdiel od neho však pridávajú podporu pre nahrávanie modulov za behu, to znamená, že nie je potrebné reštartovať celý systém. Stačí nahrať potrebné moduly, ktoré sa pripoja do jadrového adresového priestoru. Termín monolitické jadro býva niekedy nesprávne používaný pre jadro bez podpory nahrávania modulov.

Pri nahrávaní modulu dochádza k istému časovému oneskoreniu, ktoré je však minimálne.

MacOSX = hybridne jadro = kombinuje vlastnosti monolitickeho jadra a mikrokernelu

Forum: Jak tomu reálně je: Jde o dvě různá členění.

3.a) Jednak jádro (prakticky vše běží v prostoru jádra, nepotkáte ovladač v

userspace) vs. mikrojádro (jádro umí akorát pouštět procesy, přidělovat jim

paměť a předávat vzkazy).

3.b) Čistý monolit (nikdo krom původních autorů se v tom nevyzná, nelze

rozšiřovat); vrstvená architektura (komunikace jen s bezprostředně souvisejícími vrstvami); modulární architektura (lze přidávat subsystémy a poměrně snadno rozšiřovat kód).

Nu a pak máme obdobné principy aplikovatelné ke všemu ještě na celý OS (toto rozlišení řada z vás vůbec neaplikovala).

1) Jakou strukturu a jaké vlastnosti očekáváte od **vnitřní paměti** počítače

Uchovavanie dat. Priamo adresovatelna procesorom riadok a stlpec. Jej velkost je urcena schopnostou adresacie procesora -> najmensia adresovatelna jednotka – najcastejsie byte. Maximalna mozna velkost je 2^n kde n je pocet adresovych bitov procesora. Cyklus pamate = doba vyjadrujuca bud vystavenie alebo zapis informacie z/do pamate.

Staticka / dynamicka, volativita, rychla/pomala, draha/lacna

Musi mat schopnost vzpamatania sa po nejakej prevedenej operacii, prekladanie pamati

Vnutorne pamate pocitaca tvoria viacere druhy pamate, vyhodou vnutornej pamate je ze je neporovnatelne rychlejsia – nizsia pristupovana doba – ako vonkajsie pamate

Typy: RAM – random access memory – pamat s priamym pristupom, operacna pamat, rychlejsia ako pevny disk, pomalsia ako registre ci cache, energeticky zavisla. Po zapnuti pocitaca sa do nej uklada OP, spustane programy, udaje, s ktorymi prave pracujeme

ROM – read only memory, raz sa naprogramuje, energeticky nezavisla, nachadzat sa tam moze zakladny program OP = BIOS -> rutiny vstupu a vystupu -> riadenie toku udajov medzi HW a OS

CMOS – mala pamat, nizka energeticka spotreba, su v nej uchovane informacie o konfiguracii pocitaca

Register – sucast procesora, najrychlejsia, najmensia, energeticky zavisla

Cache – vyrovnavacia pamat – sucast procesora, DRAM -> periodicke obnovovanie = refresh

Pevny disk – vonkajsia pamat, najpomalsia, energeticky nezavisla, pristup pomocou kombinacie priameho a sekvencneho pristupu (-> na porovnanie)

Forum: dva různé pohledy - jak na zapojení ram, tak na hierarchii \_\_vnitřních\_\_ pamětí. Podstatný, pro vnitřní paměti, je ten fakt, že jsou v principu přímo adresovatelné procesorem, bez nutnosti nejprve nakopírovat data do RAM. Takže periferní paměti, jako třeba pevný disk sem nepatří. Takže ani odpovědi, popisující vlastnosti vnitřních a následně vnějších pamětí.

3) Popište stručně organizaci souborů pomocí **inodes**. úrovně zanoření a jak jsou propojeny.

Podobne indexovej organizacii ( tabulka pre vsetky bloky na disku ako FAT –je ulozena na zaciatku disku) ale pre kazdy subor je vlastna tabulka s pevnou dlzkou, kratsie subory su adresovane priamo

Tabulky hierarchicky previazane

Flexibilne mala rezia.

Z fora:

Disk se rozdělí na bloky pevné velikosti - např. 4KB. Tím pádem lze považovat disk za pole těchto bloků - každý blok má svůj index v tomto "globálním poli". A teď: Některé bloky ukládají přímo data souboru, některé soupisku bloků, které obsahují data souboru (tabulka). Myšlenka za inodes spočívá v následujícím:

1) Pokud je soubor dostatečně malý, ukládej rovnou jeho data (vyžaduje

samozřejmě nějakou režii okolo, aby nebyla data souboru interpretována jako

tabulka a naopak).

2) Pokud je soubor větší, tak jeho první inode obsahuje soupisku (která je různě velká podle velikosti onoho bloku). Každopádně platí, že obsahuje indexy bloků s daty souboru (tzv. přímé odkazy). 1. záznam od konce odkazuje další inode, který obsahuje jen přímé odkazy (tzv. 1x nepřímý záznam). 2. záznam od konce vede na inode, obnášející (pouze) nepřímé záznamy (tzv. 2x nepřímý záznam). A 3. záznam od konce - už si domyslete sami.

Ine struktury na disku: suvisla, linearny zoznam, indexove systemy(FAT)

4) Stručně diskutujte, jak se liší podpora **multimediálních přenosů** ve

spojovaných a přepínaných sítích. A co jsou multimediální přenosy a proč v

jejich případě vyžadujeme speciální podporu od přenosové sítě?

Multimedialne prenosy su prenosy zvuku aj obrazu po sieti

tyto přenosy vyžadují nízkou latenci -> vcasne dorucenie, neprilis velky rozptyl dorucenia paketov, vysokou propustnost a stabilitu sítě.

RTP, RTCP aplikacie

Spojovane siete: jednoduchsie riesenie, nedostatocna koncova kapacita, potencialne plytvanie pasmom

Prepinane siete: dobře vyuzitelny multicast, pozaduje sa ista kvalita služby -> pozaduje sa rezervacia : overprovision/ dedikovane okruhy/ rezervacia pre každý tok zvlast/ agregacia tokov – staticka rezervacia pre agregaciu (pre sucasny internet najvhodnejsie)

11. Jaké základní požadavky klade na **síť přenos multimediálních dat** (tedy zvuku a obrazu)?

Zvacsa vyzadujeme dostatocnu kapacitu (sirku pasma), idealne malu chybovost

(nemusi byt uplne bezchybny provoz, ale prilisna chybovost vadi). Malu

odozvu. Maly rozptyl oneskorenia (jitter). co s tym suvisi -> kvalita sluzby,rezervace.

6) Jak je v grafických systémech **zobrazována barva** (a jak ji popisujeme v SW). Uveďte jeden způsob **kódování barvy**.

Farba je schopnost povrchu nejakej hmoty odrazat alebo pohlcovat svetlo, v pocitaci je reprezentovana viacerymi sposobmi RGB, CMY, CMYK -> hovoria nam ako s farbami mame pracovat

* RGB = je aditivna - pricitava farby k bielej -> odpoveda skladaniu svetla LCD/CRT
* CMY = je substraktivna – odcitava farby od bielej -> miesanie farieb napr v tlaciarniach
* HVS model = kombinuje jas, odtien, sytost – treba k nemu pridruzit farebny priestor

7. Co je **autentizace**? A co je to **autorizace**? Uveďte jednu autentizační

Autentizacia je proces overenia identity subjektu na zaklade niecoho co vlastni/pozna/je

Autorizacia je udelenie pristupu k nejakej sluzbe na zaklade autentizacie – overenie identity napr login&heslo

Delegacia = splnomocnenie -> Korektne: vydame novy certifikat, ktory podpiseme -> entita sa preukaze tymto novym -> druha strana vidi nas podpis pod delegaciou a preto akceptuju

10. Co je to **TCP** (Transmission Control Protocol), kde a na co se používá a jaké má vlastnosti?

Forum : TCP je protokol transportnej vrstvy ISO/OSI (Transportni), jedna sa o protokol vytvarajuci spolahlivu sluzbu nad best-effort = IP, a to tak ze spojenie nadviaze (3way handshake (syn, syn+ack, ack) ), pri posielani samotnych dat su tieto potvrdzovane (doruci vsetky pakety), riadi zahltenie/tok je schopny spomalit odosielanie a tym nezahltit prijemcu/nezabrat vsetok provoz.

Na konci je spojenie ukoncene ako 1->2: FIN, 2->1:

ACK, 2->1: FIN, 1->2: ACK. Spojenie je teda ukoncene z oboch stran. (tie flagy vediet nemusite ani pri nadviazani ani pri ukonceni spojenia)

12. Jakým způsobem pomáhají informační technologie průmyslu?

1.Popište **strukturu adres** v Internetu. Víte, jak souvisí adresa a jméno uzlu? Umíte z adresy zjistit jméno uzlu (jak to uděláte)?

IP adresy jednoznacne urcuju uzol pomocou ciselnej adresy

* IPv4 – 32b – bezne sa zapisuje dekadicky A.B.C.D zlava hierarchicky
* IPv6 – 128b – rychle vycerpanie IPv4, je rozsiritelny (rozsiritelne hlavicky), podpora prenosov v realnom case, podpora zabezpecenia prenosu, podpora mobility domacich agentov, podpora autokonfiguracie

-deli sa na dve casti: cast urcujuca siet a cast urcujuca stanicu v podsieti

-unicast, multicast, broadcast - ?? asi netreba

DNS = sluzba na preklad domenovych mien na IP adresy a naspat (pomocou specialnej domeny in-adrr.arpa)

12. Jak muze informatika pomoci vedeckemu badani?

4) Čím se podle Vás liší **kernel** operačního systému od **mikrokernelu**? Proč (v jaké souvislosti) se o mikrokernelech uvažuje? V čem je možná výhoda mikrokernelu?

Kernel = jadro operacneho systemu, je zavedene do operacnej pamati hned po starte a je mu prenechane riadenie. Zodpoveda za uplne zakladne funkcie – alokacia pamate, sprava suborov, priame ovladanie hardwaru, bezpecnost – poskytuje funkcie zbytku OS, obsahuje viacej funkcionality ako mikrokernel napr integrita ovladaca

Mikrokernel – modularny pristup, male moduly zodpovedaju za funkcie, minimalizovane na nevyhnutne operacie, nezbytne minimum – ochrana pristupu namiesto celej spravy suborov. Mnoho funkcii az v uzivatelskom priestore. Vyhodou je lepsia schopnost udrzat pocitac v chode aj potom co v nejakom podsysteme dojde k zavaznej chybe – restartuje sa len podsystem. Flexibilne, upravenie OS podla potreby. Dalsia funkcionalita sa pridava pomocou modulov -> nizsi vykon ale vyssia stabilita a bezpecnost

6) Popište alespoň dva problémy, které vznikají při **zobrazení** (úseček nebo ploch apod. ) na **rasterových displejích**.

1. rozpad priamky alebo hrany nejakeho polygonu -> „schodovy efekt“

Prevedenie usecky s istym uhlom do rastru

Pozdlz danej usecky sa v krokoch po ose x pocita najblizsi pixel v ose y -> pomocou funckie round() -> neefektivne -> pouzitie Bresenhamovho algo -> vyuziva celociselne operacie bitoveho posunu

2. vypln ploch – neni presne definovany pojem oblasti (vsetky pixle danej farby/ pomocou vzdialenosti od pixla/ definovane polygonom) a pojem susedneho pixla (4-smerna = hrana/ 8-smerna definicia = vrchol/hrana)

12. Nova ekonomika

5) Co si představujete pod pojmy **úzce propojený systém a volně propojený** systém?

Který z nich využívá sdílenou paměť jako svůj typický výpočetní model? V jaké vlastnosti (aspektu) jsou volně propojené systémy lepší než ty úzce propojené?

Uzko prepojene systemy: mavaju spolocnu operacnu pamat (co moze byt vyhodne oproti posielaniu sprav ale stale moze priniest problemy). Je tam minimalny vplyv vzdialenosti procesorov -> vyhoda. Vhodne pre jemny paralelizmus. Typicky vypocetny model je zdielana pamat (aj keby mala byt iba virtualna)

Volne prepojene : distribuovana pamat – kazdy procersor zvlast(existencia prikazov remote get a remote put = pristup do pamate vzdialenych procesorov) specialne prepojenie procesorov, ich vzdialenost moze hrat rolu, vyssia latencia ako v medziprocesorovej komunikacii. Vyhodou je, ze sa daju lahko a lacnejsie rozsirovat. Typickym vypocetnym modelom je posielanie sprav.

Forum:Nejčastejšou chybou bolo to, že niektori z vás považujú za výhodu voľne prepojených systémov to, že nemajú zdieľanú pamäť. Zdieľaná pamäť síce prináša pár problémov ako s ňou pracovať, ale je omnoho rýchlejšia oproti druhému modelu -- posielaniu správ. Jedna zo správnych odpovedí na túto podotázku je napríklad to že voľne porepojený systém sa dá ľahsie a lacnejšie rozširovať.

6) Co si představujete pod pojmem barevný **gamut**?

Mnozina farieb ktorymi zariadenie disponuje -> vysek farebneho priestoru -> vie zobrazit iba urcite farby farebneho priestoru

„Co je to **Certifikační autorita** a jakou má roli? Certifikační autorita patří obecně ke konceptu symetrické nebo asymetrické kryptografie?“

* Priradenie verejneho kluca ku konkretnej entite

CA je institucia ktora overi kto je vlastnik sukromneho kluca k urcitemu verejnemu -> vyda certifikat potvrdzujuci tuto vazbu

V asymetrickej kryptografii je to subjekt, ktory vydava digitalne certifikaty

Co je **TCP**?

Smerovaci protokol transportnej vrstvy. Poskytuje istotu dorucenia paketu (potvrdzovanie dorucenia, v pripade nedorucenia sa posle znovu), vie korigovat mnozstvo posielanych dat za jednotku casu -> pri zahlteni prijemcu spomali, pouziva ho vacsina prenosov internetu -> prenos suborov, emailov, www

Pocas komunikacie sa udrzuje spojenie medzi odosielatelom a adresatom, rozdeluje udaje do segmentov

Jeho „protejskom“ je UDP, ktore sa pouziva napr u livestreamu, je rychlejsie lebo nie je potrebne dorucit vsetky pakety

+ viz otazka o TCP vyssie

7) Jak **jsou reprezentována v počítači "reálná" čísla** (tedy čísla s desetinnou částí)? Na které části má vliv přesnost zobrazení?

Realne cisla mozu byt statickou alebo pohyblivou desatinnou ciarkou.

So staticou su zobrazene cele aj desatinne casti urcitym poctom bitov. Format : <znamienkovy bit><rady s kladnym exp>,<rady so zapornym exp>

Realne cisla s pohyblivou desatinnou ciarkou – podla standardu IEEE754 :

<znamienkovy bit><8 bitov – exponent><mantisa 23b> vysledne cislo je znamienko, mantisa \* 2^exponent

IEEE 754 podrobnejsie : Pozname viacero formatov (binary32, binary64,...). Prvy bit je znamienkovy, potom exponent v kode posunutej nuly (ktory nam udava o kolko desatinnych miest sme sa posunuli a taktiez udava rozsah zobrazenia) a mantisa (cislo v normalizovanom tvare) – nalavo od desatinnej bodky je len jednotka, doplnene nulami na potrebny pocet bitov.

Presnost cisla ovplyvnuje len mantisa.

Forum: Obecně se u této otázky pouze stačilo zamyslet, napsat vzoreček (správně) a okomentovat jej. Odpověď na druhou část otázky pak máte uvedenou výše. Jakékoliv smysluplné doplňující informace pak byly jen k dobru.

7) K čemu slouží **vyrovnávací paměť** u systému souborů? Diskutujte rozdíly v

případě použití jen pro čtení versus využití i pro zápis. Jaké základní typy vyrovnávacích pamětí znáte?

Vyrovnávacia pamäť alebo hovorovo zásobník ([angl.](https://sk.wikipedia.org/wiki/Angli%C4%8Dtina) buffer) je spravidla vyhradená časť [pamäte](https://sk.wikipedia.org/wiki/Pam%C3%A4%C5%A5_(po%C4%8D%C3%ADta%C4%8D)) určená na dočasné ukladanie dát pred spracovaním (na vyrovnanie rýchlostí odosielajúcej jednotky a prijímacej jednotky alebo na prevenciu rôznych výpadkov pri prenose dát); v [tlačiarňach](https://sk.wikipedia.org/wiki/Tla%C4%8Diare%C5%88_(hardv%C3%A9r)) a niektorom inom [hardvéri](https://sk.wikipedia.org/wiki/Hardv%C3%A9r) to môže byť aj špeciálna malá pamäť s podobnou funkciou.

Rýchla vyrovnávacia pamäť ([angl.](https://sk.wikipedia.org/wiki/Angli%C4%8Dtina) cache) je malá, ale rýchla [vyrovnávacia pamäť](https://sk.wikipedia.org/wiki/Vyrovn%C3%A1vacia_pam%C3%A4%C5%A5), ktorá slúži na prechodné ukladanie dát alebo sekvencií programu.

Vyrovnávacie pamäte sa používajú všade, kde môžu zvýšiť výkonnosť [hardvéru](https://sk.wikipedia.org/wiki/Hardv%C3%A9r) alebo informačného systému, napríklad v [pevných diskoch](https://sk.wikipedia.org/wiki/Pevn%C3%BD_disk), [mikroprocesoroch](https://sk.wikipedia.org/wiki/Mikroprocesor), [základných doskách](https://sk.wikipedia.org/wiki/Mati%C4%8Dn%C3%A1_doska), ako aj v [operačných systémoch](https://sk.wikipedia.org/wiki/Opera%C4%8Dn%C3%BD_syst%C3%A9m).

Základné delenie:

* hardvérová rýchla vyrovnávacia pamäť: je realizovaná osobitnými pamäťovými čipmi
* softvérová rýchla vyrovnávacia pamäť: je realizovaná v časti operačnej pamäte, kde sa dočasne ukladajú častejšie používané dáta

Existuju 3 hlavne techniky ukladania:

* write-through – ulozi I/O priamo na cache a potom na pevny disk, nasledne to potvrdi
* write-around – podobne jako write-through ale najprv zapise do permanentnej pamate a potom premosti do cache
* write-back – ulozi I/O na cache a potvrdi to hned -> nizsia latencia oproti write-through. Iba kopia dat je na cahce -> risk
  + zakladny rozdiel write-through okamzite zapise aj na disk a write-back az po urcitej době (napr 30 s)

Forum : Část z vás psala o žurnálování, značná část o cache procesoru. Otázka byla zaměřena jednak poněkud obecněji, jednak neřešila obnovu při výpadku. Málo z vás zmínilo (a osvětlilo) pojmy jako write-back nebo write-through.

7)Co rozumíte pod pojmem **renderování**? Jak tomuto procesu pomáhají grafické

procesory (GPU)? Co dnes od GPU očekáváte, že budou umět?

je to vykreslení obrázku z počítačového modelu. Vyuzivaju sa u toho rozne techniky ako raytracing, Z-buffering, texturovanie... Pri renderovani offline sa da vytvorit pomerne realisticke videa a obrazky -> rendrovanie vtedy ale moze trvat aj niekolko minut. Ak chceme renderovat v realnom case musime znizit nase naroky na kvalitu aby sme dosiahli aspon 25fps.

Forum : A krom něj to, že jej budeme moci využit jako GPGPU. Vysoký výpočetní výkon se očekává takřka od veškeré techniky.

GPU = specializovany procesor grafickej karty sluziaci na vypocet grafickych informacii pre zobrazenie udajov pocitaca. Prevadza digitalne data do formy zobrazitelnej pomocou obrazkovych pixelov. Ulohov su aj prepocty virtualnych trojrozmernych scen v realnom case. Vykonava vsetky graficke instrukcie a tym setri strojovy cas mikroprocesora pocitaca

GPGPU = programovatelne, daju sa vyuzit aj na vedecke vypocty -> graficka karta reaguje ako stream procesory

OpenGL = Open graphics Library

* Je Api medzi programatorom a grafickym HW -> specifikuje subor funkcii, ktore umoznuju vykreslovanie zakladnych 2D a 3D obejktov, texturovanie, osvetlenie, tienovanie (nepodporuje tiene)

5) Srovnejte principy **autentizace** pomocí **symetrické a asymetrické kryptografie**. Který z těchto dvou přístupů používá delší klíče?

Forum:

Pro symetrickou kryptografii stačí vzít v úvahu následující schéma:

1) Strana A vygeneruje náhodné číslo, provede nad ním dohodnutou operaci č. 1 a jak číslo, tak výsledek operace zašifruje klíčem.

2) Tuto zprávu odešle straně B.

3) Strana B zprávu dešifruje a nad číslem provede dohodnutou operaci č. 2. Ověří si současně výsledek dohodnuté operace č. 1, který poslala strana A. Tím si ověří, že strana A zná klíč (jinak by výsledek neodpovídal).

3) Strana B zašifruje nový výsledek klíčem a pošle jej straně A.

4) Strana A ověří výsledek B. Toto autentizační schéma dovoluje autentizovat jak stranu A, tak stranu B – obě musí znát klíč, aby se dobraly správným výsledkům. Často jste tvrdili, že stačí když dohodnutou operaci provede akorát strana B. Toto ale neautentizuje stranu

A, neboť za A může vystupovat Zlosyn (Z), který si vymyslí náhodnou posloupnost bytů, tu prohlásí za zašifrovanou zprávu, nechá B spočítat výsledek a ten bez ohledu na správnost akceptuje. B by si tak myslelo že Z je A a zná klíč, aniž by to byla pravda.

U asymetrické kryptografie bylo potřeba popsat jak funguje mechanismus

digitálního podpisu. Tedy to, že strana A vytvoří nějaká data, ta (popř. Jejich hash) zašifruje soukromým klíčem a obojí vystaví. Strana B pak má k dispozici veřejný klíč a může si ověřit, že zašifrovaná data po dešifrování veřejným klíčem odpovídají nezašifrovaným -- tedy B prokáže znalost soukromého klíče. Tento autentizační mechanismus je jednostranný, neboť strana B se neprokazuje vůbec. Také se nám problém autentizace redukuje na problém distribuce veřejných klíčů.

Dlhsie kluce pouziva asymetricka (napr.2048/4096)/symetricka (napr.128/256)

4) Co je to **ad-hoc** síť a na jakých principech je postavena? Kde byste se s ní mohli setkat, případně jaké má využití?

Ad-hoc siet je decentralizovana siet s kratkym dosahom vysielaca vyuzivana najma pri zivelnych katastrofach -> siet bez infrastruktury

Pouziva sietove vlastnosti ucastnikov, nema k dispozicii access pointy (routre) -> kazdy uzol je smerovac a zaroven aj koncovy bod, potreba dynamickej topologie siete -> riadenie rozdelene medzi jednotlive uzly

Vyhody: rychle vybudovanie, odolnost, efektivne vyuzitie radioveho spektra

Nevyhody: obmedzeny dosah, komplikovane riadenie, zmeny v topologii pri pohybe uzlov.

Forum: decentralizovaných sítích s krátkým dosahem vysílače (řekněme

max. dosah 2 uzly). Pak už jen stačilo popsat vlastnosti které musíte přidat aby tato síť fungovala (tedy client/relay, routovací schéma, pravidla expanze sítě). Pak už jen najít vhodný příklad použití - třeba senzorovou síť nebo tým hasičů s vysílačkami uprostřed trosek Fukušimy IV (asi nejlepší příklad -- beton budov odstiňuje signál). Dva notebooky vyměňující si soubor zrovna mezi nejlepší příklady nepatří.

Dufam, ze aspon trochu pomoze. Vela stastia na skuske ☺