STŘEDNÍ PRŮMYSLOVÁ ŠKOLA A VYŠŠÍ ODBORNÁ ŠKOLA, PÍSEK, KARLA ČAPKA 402

Maturitní témata 2025/2026

Využití technického a programového vybavení. Pracovní podklady pro ústní maturitní odpověď přibližně na 15 minut ke každému tématu.

Jak s podklady pracovat

Každé téma je stavěné jako osnova pro mluvený výklad, ne jako text k mechanickému čtení. Nejlepší postup je naučit se pořadí myšlenek, doplnit vlastní příklady a při zkoušení mluvit souvisle: definice, princip, důležité vlastnosti, praktické použití, závěrečné shrnutí.

U technických témat se vyplatí kreslit jednoduché blokové schéma. U programování je dobré přidat krátký příklad v libovolném známém jazyce. U databází, Git a webových technologií pomáhá ukázat typický postup z praxe.

Doporučené časování odpovědi

- **1 minuta:** vymezení tématu a základní pojmy.
- **4 minuty:** hlavní principy a členění.
- **6 minut:** podrobnější vysvětlení nejdůležitějších částí.
- **3 minuty:** příklady, parametry, srovnání nebo praxe.
- **1 minuta:** shrnutí a návaznosti na další témata.

Obsah

5. Architektury počítačů, instrukční cyklus a sběrnice
6. Paměti
7. Pevné disky, SSD a RAID
8. Základní deska, firmware a rozhraní PC
9. Grafika, zobrazovače, projektory a zvuk
10. Tiskárny, barevné modely, skenery a OCR
11. Vývoj počítačů a operačních systémů
12. Přidělování paměti
13. POST, multitasking a přerušení
14. Synchronizace procesů
15. Souborové systémy
16. Algoritmy a neuronové sítě
17. Běh programu a přenositelnost
18. Vývojové diagramy, UML a dokumentace
19. Relační databáze a SQL: CRUD, integrita, indexy, transakce
20. Relační vazby a JOIN
21. Git
22. Skalární typy, proměnné a výrazy
23. Řídicí a datové struktury
24. Výjimky, chyby a logování
25. Funkce
26. OOP: třídy, objekty a členy
27. OOP: dědičnost, rozhraní a polymorfismus
28. REST API
29. Klient-server, HTTP, AJAX a cookies
30. Bezpečné chování online

---

1. Logické obvody, pravdivostní tabulka, algebraický výraz, zákony Booleovy algebry, multiplexor, porovnávací obvod

kombinační logika | AND OR NOT | minimalizace | MUX

Osnova odpovědi

1. Vysvětli, že logický obvod pracuje s hodnotami 0 a 1, tedy se dvěma stavy napětí.
2. Rozliš kombinační obvody, kde výstup závisí jen na aktuálních vstupech, a sekvenční obvody, kde hraje roli paměť.
3. Popiš základní hradla NOT, AND, OR, NAND, NOR, XOR a jejich pravdivostní tabulky.
4. Ukaž převod mezi slovním zadáním, pravdivostní tabulkou, algebraickým výrazem a schématem z hradel.
5. Přidej zákony Booleovy algebry a význam zjednodušování obvodů.
6. Vysvětli multiplexor a porovnávací obvod jako praktické kombinační obvody.

Výkladové body

Booleova algebra používá proměnné, které nabývají jen hodnot 0 nebo 1. Operace součin, součet a negace odpovídají logickým funkcím AND, OR a NOT. Výraz `Y = A · B + /C` říká, že výstup bude jednička, pokud platí současně A i B, nebo pokud není C. Stejnou funkci lze popsat pravdivostní tabulkou, kde se vypíší všechny kombinace vstupů a výsledný výstup.

Důležité zákony jsou komutativní, asociativní, distributivní, zákon identity, nulový zákon, zákon idempotence, zákon dvojí negace a De Morganovy zákony. Ty umožňují zjednodušit výraz, snížit počet hradel a tím i cenu, spotřebu a zpoždění obvodu.

Příklady do odpovědi

- **Multiplexor:** přepínač více vstupů na jeden výstup podle adresových vstupů. Například 4:1 MUX má 4 datové vstupy, 2 adresové vstupy a jeden výstup.
- **Porovnávací obvod:** komparátor porovnává dvě binární čísla a určí, zda A = B, A > B nebo A < B.
- **NAND a NOR:** univerzální hradla, ze kterých lze sestavit libovolnou logickou funkci.

> Nezaměň pravdivostní tabulku s tabulkou stavovou. Pravdivostní tabulka popisuje kombinační logickou funkci bez paměti.

2. Sekvenční logické obvody: blokové schéma, klopné obvody, čítače, registry

paměť stavu | hodinový signál | klopné obvody | registry

Osnova odpovědi

1. Začni rozdílem mezi kombinačním a sekvenčním obvodem.
2. Nakresli blokové schéma: vstupy, kombinační logika, paměťové prvky, zpětná vazba, výstupy.
3. Vysvětli hodinový signál, synchronní a asynchronní řízení.
4. Popiš klopné obvody SR, JK, D a T.
5. Přejdi k čítačům: binární, dekadické, nahoru/dolů, synchronní/asynchronní.
6. Vysvětli registry jako skupinu klopných obvodů pro uchování nebo posun dat.

Výkladové body

Sekvenční obvod má paměť. Výstup tedy nezávisí jen na okamžitých vstupech, ale také na předchozím stavu. Typické je řízení hodinovým signálem, který určuje okamžik změny stavu. Synchronní obvody mění stav společně podle hodin, asynchronní reagují přímo na změny vstupů a mohou být citlivější na zpoždění.

Klopný obvod je základní paměťový prvek pro jeden bit. D klopný obvod přenese hodnotu ze vstupu D na výstup při aktivní hraně hodin. T klopný obvod při aktivním taktu překlápí stav. JK klopný obvod odstraňuje zakázaný stav známý z jednoduššího SR obvodu.

Praktické použití

- **Čítače:** počítání impulsů, děliče frekvence, časovače, adresování paměti.
- **Registry:** uchování instrukce, dat, adresy nebo stavového slova v procesoru.
- **Posuvné registry:** převod paralelních dat na sériová a naopak.

> Dobrá zkoušková věta: sekvenční logika je základ procesorů, pamětí, komunikačních rozhraní i řídicích automatů.

3. Architektura mikroprocesoru CPU, blokové schéma, popis bloků CU, RI, Di, ALU, PSW

CPU ALU řadič registry

Osnova odpovědi

1. Definuj CPU jako hlavní výpočetní a řídicí jednotku počítače.
2. Nakresli jednoduché blokové schéma CPU a propojení s pamětí a sběrnicemi.
3. Popiš řadič CU, registr instrukce RI, dekodér instrukcí Di, ALU a PSW.
4. Vysvětli obecné registry, programový čítač a zásobníkový ukazatel.
5. Uveď fáze instrukčního cyklu: načtení, dekódování, vykonání, zápis výsledku.
6. Zmiň výkonové parametry: takt, počet jader, cache, šířka slova, instrukční sada.

Výkladové body

Mikroprocesor je integrovaný obvod, který provádí instrukce programu. Řadič CU řídí tok dat a vydává řídicí signály ostatním blokům. RI uchovává právě zpracovávanou instrukci. Dekodér instrukcí Di určí, jaká operace se má provést a s jakými operandy. ALU provádí aritmetické a logické operace, například sčítání, odčítání, porovnání, bitové AND nebo posuny.

PSW, tedy program status word, obsahuje příznaky po operaci. Typicky jde o zero, carry, sign, overflow nebo parity flag. Tyto příznaky se využívají při podmíněných skocích a řízení toku programu.

Co nezapomenout

- CPU samo o sobě obvykle neobsahuje operační paměť programu, komunikuje s ní přes sběrnice.
- Cache paměť zmenšuje rozdíl mezi rychlostí procesoru a hlavní paměti.
- Výkon neurčuje jen frekvence, ale také architektura, počet jader, IPC, cache a optimalizace programu.

> Nepleť RI jako registr instrukce s běžným datovým registrem. RI drží instrukci, kterou řadič právě dekóduje.

4. Architektura mikrokontroleru MCU, sběrnice, popis bloků paměť, I/O obvody, C/T

MCU SoC I/O časovače

Osnova odpovědi

1. Vysvětli rozdíl mezi mikroprocesorem a mikrokontrolerem.
2. Popiš MCU jako čip obsahující CPU, paměti a periferie.
3. Vysvětli interní sběrnice: datovou, adresovou a řídicí.
4. Prober paměti programu, RAM a případnou EEPROM nebo flash pro data.
5. Popiš I/O porty, digitální vstupy/výstupy, analogové vstupy, PWM, UART, SPI, I2C.
6. Vysvětli C/T, tedy čítače a časovače, a jejich použití.

Výkladové body

Mikrokontroler je určen hlavně pro vestavěné systémy. Na jednom čipu bývá procesorové jádro, paměť programu, datová paměť, vstupně-výstupní obvody, časovače, komunikační rozhraní a často i A/D převodník. Díky tomu může řídit zařízení bez složité externí elektroniky.

Paměť programu je často typu flash, takže firmware zůstane uložen i po odpojení napájení. RAM slouží pro proměnné za běhu programu. I/O obvody připojují MCU k okolnímu světu: tlačítka, LED, senzory, relé, displeje nebo komunikační moduly. Časovače a čítače umožňují měření času, generování PWM, počítání impulsů nebo vyvolání přerušení.

Příklady

- Arduino, ESP32, STM32, AVR nebo PIC jsou typické platformy s mikrokontrolerem.
- PWM lze použít k řízení jasu LED nebo otáček motoru.
- UART, SPI a I2C jsou běžná rozhraní pro senzory a moduly.

> Silná maturitní odpověď porovná MCU a CPU: MCU je úspornější a integrovanější, CPU ve stolním PC je výkonnější a spoléhá na externí čipovou sadu a paměti.

5. Von Neumannova a Harvardská architektura, RISC a CISC, instrukční cyklus, zřetězení instrukcí, taxonomie sběrnic, paralelní, sériový, synchronní a asynchronní přenos dat

architektury pipeline sběrnice přenos dat

Osnova odpovědi

1. Porovnej Von Neumannovu a Harvardskou architekturu.
2. Vysvětli rozdíl mezi RISC a CISC.
3. Popiš instrukční cyklus a jeho fáze.
4. Vysvětli zřetězení instrukcí jako překrývání fází více instrukcí.
5. Uveď druhy sběrnic podle funkce, směru, šířky a způsobu přenosu.
6. Porovnej paralelní/sériový a synchronní/asynchronní přenos dat.

Výkladové body

Von Neumannova architektura používá společnou paměť pro instrukce i data, takže program i data sdílejí stejné paměťové rozhraní. Harvardská architektura má paměť instrukcí a paměť dat oddělenou, což umožňuje současně načítat instrukci a pracovat s daty. Mnoho moderních procesorů používá upravený hybrid: navenek jednotná paměť, uvnitř oddělené cache pro instrukce a data.

CISC má bohatší a často složitější instrukční sadu, typicky s instrukcemi proměnné délky. RISC preferuje jednodušší instrukce, pevnější formát, práci s registry a snadnější zřetězení. Pipeline zvyšuje propustnost tak, že jedna instrukce se načítá, druhá dekóduje a třetí provádí. Problémem jsou skoky, závislosti dat a konflikty zdrojů.

Sběrnice a přenosy

- **Adresová sběrnice:** vybírá místo v paměti nebo zařízení.
- **Datová sběrnice:** přenáší data.
- **Řídicí sběrnice:** přenáší signály čtení, zápisu, přerušení a potvrzení.
- **Paralelní přenos:** více bitů současně, kratší vzdálenosti.
- **Sériový přenos:** bity za sebou, jednodušší kabeláž, dnes velmi rozšířený.
- **Synchronní přenos:** společné hodiny. **Asynchronní:** domluvený formát, start/stop bity nebo handshake.

6. Paměti: rozdělení, realizace, statické a dynamické paměti, organizace buněk, prokládání cyklů

RAM ROM SRAM DRAM cache flash

Osnova odpovědi

1. Definuj paměť jako zařízení pro uchování instrukcí a dat.
2. Rozděl paměti podle přístupu: sekvenční, přímý, náhodný, asociativní.
3. Rozděl je podle zápisu: ROM, PROM, EPROM, EEPROM, flash, RAM.
4. Vysvětli volatilní a nevolatilní paměti.
5. Porovnej SRAM a DRAM.
6. Prober organizaci paměťových buněk, adresování, šířku slova a prokládání cyklů.

Výkladové body

RAM je paměť s náhodným přístupem, používaná pro běh programů. Je volatilní, takže po vypnutí napájení obsah ztratí. ROM a flash jsou nevolatilní, proto se hodí pro firmware, BIOS/UEFI, SSD nebo paměťové karty. Podle účelu se paměti dělí na registry, cache, operační paměť, vnější úložiště a záložní paměti.

SRAM uchovává bit pomocí klopného obvodu. Je rychlá, drahá a zabírá více místa, proto se používá hlavně jako cache. DRAM uchovává bit jako náboj v kondenzátoru, musí se obnovovat refreshem, ale je levnější a hustší, proto tvoří hlavní operační paměť. Synchronní DRAM pracuje podle hodinového signálu a umožňuje efektivní přenosy v dávkách.

Pojmy k doplnění

- **Adresace:** procesor vybírá buňku pomocí adresy.
- **Organizace:** například počet řádků, sloupců, bank a šířka datové sběrnice.
- **Prokládání cyklů:** střídání více paměťových bank tak, aby se překrylo čekání a zvýšila propustnost.
- **Latence a propustnost:** latence je doba čekání, propustnost množství dat za čas.

> Častý omyl: SSD není operační paměť. Je nevolatilní úložiště založené na flash paměti.

7. Pevné disky, fyzická a logická struktura, magnetický zápis a čtení, SSD, parametry disků, RAID

HDD SSD RAID úložiště

Osnova odpovědi

1. Vysvětli rozdíl mezi HDD a SSD.
2. Popiš fyzickou strukturu HDD: plotny, stopy, sektory, cylindry, hlavy.
3. Vysvětli magnetický zápis a čtení.
4. Popiš logickou strukturu: oddíly, souborový systém, bloky, LBA.
5. Uveď parametry: kapacita, otáčky, cache, přístupová doba, rychlost, rozhraní, spolehlivost.
6. Vysvětli SSD a základní úrovně RAID.

Výkladové body

HDD ukládá data magneticky na rotující plotny. Čtecí a zapisovací hlavy se pohybují nad povrchem ploten. Data jsou organizována do sektorů a moderně adresována pomocí LBA, tedy logických blokových adres. Přístupová doba je ovlivněna seek time, tedy přesunem hlavy, a rotační latencí, tedy čekáním na správné natočení plotny.

SSD nemá mechanické části. Data jsou uložena ve flash pamětech, proto má mnohem nižší latenci, vyšší počet náhodných operací za sekundu a lepší odolnost proti otřesům. Má ale omezený počet zápisových cyklů, který se řeší wear levelingem, rezervními bloky a řadičem SSD.

RAID v kostce

- **RAID 0:** striping, vyšší výkon a kapacita, bez redundancy.
- **RAID 1:** mirroring, data jsou zrcadlena, vyšší bezpečnost.
- **RAID 5:** striping s paritou, přežije výpadek jednoho disku.
- **RAID 6:** dvě parity, přežije výpadek dvou disků.
- **RAID 10:** kombinace zrcadlení a stripingu.

> RAID nenahrazuje zálohování. Chrání hlavně dostupnost při poruše disku, ne proti smazání, viru nebo požáru.

8. Základní deska, čipová sada, BIOS, UEFI, skříně PC, rozhraní, ovládací zařízení, chlazení, UPS a zdroje

motherboard UEFI USB Ethernet WiFi PSU UPS

Osnova odpovědi

1. Popiš základní desku jako nosnou a propojovací část počítače.
2. Uveď socket CPU, sloty RAM, PCIe, konektory disků, napájení a firmware.
3. Vysvětli čipovou sadu a její dnešní roli.
4. Porovnej BIOS a UEFI.
5. Prober skříně PC, chlazení, napájecí zdroj a UPS.
6. Vysvětli rozhraní a ovládací zařízení: USB, Ethernet, WiFi, Bluetooth, IrDA, klávesnice, myš, touchpad, trackpoint.

Výkladové body

Základní deska propojuje procesor, paměť, grafickou kartu, disky a periferie. Obsahuje napájecí obvody, sloty, konektory, firmware a řadiče. Čipová sada dříve zajišťovala severní a jižní můstek; dnes je mnoho funkcí integrováno přímo v procesoru a čipset zajišťuje hlavně další linky PCIe, USB, SATA a správu periferií.

BIOS je starší firmware, který inicializuje hardware a spouští zavaděč operačního systému. UEFI je modernější, podporuje grafické rozhraní, větší disky s GPT, moduly, síťové funkce a secure boot. Skříň ovlivňuje form factor, proudění vzduchu, hlučnost, montáž a rozšiřitelnost.

Parametry a příklady

- **USB:** univerzální sériová sběrnice pro periferie a napájení.
- **Ethernet:** drátová síť, typicky 1 Gb/s a více.
- **WiFi/Bluetooth:** bezdrátové technologie, WiFi pro síť, Bluetooth pro blízké periferie.
- **PSU:** sleduj výkon, účinnost 80 Plus, konektory a ochrany.
- **UPS:** záložní zdroj chrání před výpadkem a může umožnit bezpečné vypnutí.

9. Grafické karty, grafická rozhraní, GPU, zobrazovací jednotky, projektory a zvukový podsystém PC

GPU LCD OLED HDMI DisplayPort audio

Osnova odpovědi

1. Vysvětli účel grafické karty a rozdíl mezi integrovanou a dedikovanou grafikou.
2. Popiš GPU, VRAM, výstupní rozhraní a sběrnici PCI Express.
3. Uveď parametry grafických karet.
4. Popiš CRT, LCD, PDP a OLED zobrazovače.
5. Vysvětli základní parametry monitorů a projektorů.
6. Přidej zvukový podsystém, zvukové karty a formáty.

Výkladové body

Grafická karta zpracovává obrazová data a vytváří signál pro zobrazovací zařízení. GPU je procesor optimalizovaný na masivně paralelní výpočty, například rasterizaci, shader programy, video akceleraci a výpočty v oblasti AI nebo simulací. VRAM uchovává textury, framebuffery a další grafická data.

CRT používá elektronový paprsek dopadající na luminofor. LCD pracuje s tekutými krystaly a podsvícením. OLED má samosvítící organické diody, takže nabízí výborný kontrast a černou, ale může trpět vypalováním obrazu. PDP, tedy plazmový panel, používá buňky s plynem a luminoforem, dnes už je spíše historický.

Parametry

- **Grafika:** výkon GPU, VRAM, šířka paměťové sběrnice, spotřeba, chlazení, podpora API.
- **Monitor:** rozlišení, úhlopříčka, obnovovací frekvence, odezva, jas, kontrast, barevný gamut.
- **Projektor:** technologie LCD/DLP, světelný tok v lumenech, kontrast, rozlišení, projekční vzdálenost.
- **Zvuk:** DAC/ADC, vzorkovací frekvence, bitová hloubka, počet kanálů, latence.
- **Formáty:** WAV je často nekomprimovaný, MP3/AAC ztrátové, FLAC bezeztrátový.

10. Tiskárny, barevné modely RGB a CMYK, PostScript a PCL, DPI, CPI, PPI, skenery a OCR

tisk CMYK DPI OCR

Osnova odpovědi

1. Rozděl tiskárny podle principu: jehličkové, inkoustové, laserové.
2. Popiš vlastnosti, výhody a nevýhody každého typu.
3. Vysvětli barevné modely RGB a CMYK.
4. Uveď tiskové jazyky PostScript a PCL.
5. Vysvětli DPI, CPI a PPI.
6. Popiš princip skeneru a OCR.

Výkladové body

Jehličková tiskárna je nárazová tiskárna. Jehličky přes barvicí pásku vytvářejí body na papíře. Je hlučná a pomalejší, ale hodí se pro průpisové formuláře. Inkoustová tiskárna vystřikuje kapky inkoustu na papír, má dobrý barevný tisk a nižší pořizovací cenu, ale inkoust může zasychat. Laserová tiskárna používá fotoválec, laser, toner a zapékací jednotku. Je rychlá, přesná a výhodná pro text a kancelářský tisk.

RGB je aditivní model používaný u displejů, kde se světlo skládá z červené, zelené a modré. CMYK je subtraktivní model pro tisk, používá azurovou, purpurovou, žlutou a černou. PostScript je stránkový popisný jazyk vhodný pro profesionální sazbu, PCL je běžný tiskový jazyk vyvinutý firmou HP.

Pojmy

- **DPI:** dots per inch, hustota tiskových bodů.
- **PPI:** pixels per inch, hustota obrazových pixelů.
- **CPI:** characters per inch, počet znaků na palec u textového tisku.
- **Skener:** snímá předlohu optickým senzorem, převádí světlo na elektrický signál a obrazová data.
- **OCR:** rozpoznávání znaků z obrazu do editovatelného textu.

> DPI a PPI spolu souvisejí, ale nejsou totéž: DPI je typicky fyzický tisk, PPI popisuje obraz nebo displej.

11. Vývoj počítačů, vývoj OS, počítačová platforma, dělení operačních systémů, prostředky výpočetního systému, funkce a struktura OS

historie OS platforma kernel

Osnova odpovědi

1. Naznač vývoj počítačů od elektronkových strojů přes tranzistory a integrované obvody k osobním počítačům a mobilním zařízením.
2. Vysvětli, co je počítačová platforma.
3. Popiš vývoj operačních systémů od dávkového zpracování k interaktivním, síťovým a mobilním OS.
4. Rozděl OS podle účelu, počtu uživatelů, multitaskingu a zařízení.
5. Vysvětli prostředky výpočetního systému.
6. Popiš funkce a strukturu OS.

Výkladové body

Počítače prošly vývojem od velkých sálových zařízení přes minipočítače, osobní počítače, notebooky, servery, mobilní zařízení a cloudové infrastruktury. Základní zlomy představují elektronky, tranzistory, integrované obvody, mikroprocesory a masová síťová komunikace.

Operační systém je základní software, který spravuje hardware a poskytuje služby programům i uživatelům. Řídí procesor, paměť, soubory, zařízení, síť, bezpečnost a uživatelské rozhraní. Počítačová platforma je kombinace hardwaru, instrukční architektury, operačního systému, knihoven a běhového prostředí, pro kterou je software určen.

Struktura OS

- **Jádro:** plánování procesů, paměť, ovladače, systémová volání.
- **Ovladače:** komunikace s konkrétním hardwarem.
- **Souborový systém:** organizace dat na úložišti.
- **Shell a GUI:** rozhraní pro uživatele.
- **Služby:** síť, tisk, aktualizace, bezpečnost.

> Příklady OS: Windows, Linux, macOS, Android, iOS, serverové a real-time systémy.

12. Přidělování paměti: absolutní a relativní adresa, metody přidělování, stránkování, segmentace, adresace, přerušení

adresace paging segmentace virtual memory

Osnova odpovědi

1. Vysvětli, proč OS musí spravovat paměť.
2. Rozliš absolutní a relativní adresu.
3. Popiš jednoduché metody přidělování: pevné oddíly, proměnné oddíly, first fit, best fit, worst fit.
4. Vysvětli fragmentaci a ochranu paměti.
5. Popiš stránkování a stránkovací tabulku.
6. Vysvětli segmentaci a kombinaci stránkování se segmentací.

Výkladové body

Absolutní adresa je skutečná adresa v paměti, relativní adresa se vztahuje k začátku programu, segmentu nebo jiného prostoru. Překlad adres umožňuje program spustit na různých místech paměti a zároveň chránit procesy před vzájemným přepisem.

Stránkování dělí virtuální paměť na stránky a fyzickou paměť na rámce stejné velikosti. Stránkovací tabulka mapuje virtuální stránky na fyzické rámce. Segmentace dělí program logicky na segmenty, například kód, data a zásobník, každý se základní adresou a limitem. Kombinace obou metod využívá logické segmenty a uvnitř nich stránky.

Přerušení a paměť

- Při výpadku stránky, tedy page fault, vznikne výjimka/přerušení a OS načte stránku z disku.
- MMU provádí překlad virtuálních adres na fyzické.
- TLB je rychlá cache překladů adres.
- Virtuální paměť umožňuje běh programů větších než fyzická RAM, ale při nadměrném stránkování klesá výkon.

> Fragmentace je jiná u oddílů a u stránkování: stránkování řeší externí fragmentaci, ale může mít vnitřní fragmentaci v poslední stránce.

13. POST, multitasking a systém přerušení: průběh POSTu, BIOS, UEFI, secure boot, druhy přerušení, druhy multitaskingu

POST boot interrupts multitasking

Osnova odpovědi

1. Popiš, co se děje po zapnutí počítače.
2. Vysvětli POST a jeho účel.
3. Porovnej BIOS a UEFI a zmiň secure boot.
4. Vysvětli přerušení jako mechanismus reakce na události.
5. Rozděl přerušení na hardwarová, softwarová a výjimky.
6. Popiš kooperativní a preemptivní multitasking.

Výkladové body

Po zapnutí počítače se spustí firmware, který inicializuje hardware, provede POST a najde zaváděcí zařízení. POST, Power-On Self-Test, kontroluje základní části počítače, například procesor, paměť, grafiku, klávesnici nebo úložiště. Pokud dojde k chybě, může ji signalizovat kódem, zvukem nebo hlášením.

UEFI je modernější náhrada BIOSu. Secure boot ověřuje digitální podpisy zavaděče a pomáhá bránit spuštění neoprávněného kódu před startem OS. Přerušení umožňuje procesoru přestat dočasně vykonávat běžný program a obsloužit událost, například vstup z klávesnice, časovač nebo chybu.

Multitasking

- **Kooperativní:** proces se musí sám vzdát procesoru.
- **Preemptivní:** OS přiděluje časové kvantum a může proces přerušit.
- **Proces:** běžící program se samostatným adresním prostorem.
- **Vlákno:** lehčí jednotka běhu uvnitř procesu.

> Časovač generuje pravidelná přerušení, díky kterým může OS plánovat procesy a vytvářet dojem souběžného běhu.

14. Synchronizace procesů: kritická sekce, producent-konzument, čtenáři a písaři, pět hladových filozofů

concurrency mutex semafor deadlock

Osnova odpovědi

1. Vysvětli, proč souběžné procesy potřebují synchronizaci.
2. Definuj kritickou sekci a závodní podmínku.
3. Uveď synchronizační prostředky: zámek, mutex, semafor, monitor.
4. Popiš problém producent-konzument.
5. Popiš problém čtenářů a písařů.
6. Popiš problém pěti hladových filozofů a riziko deadlocku.

Výkladové body

Kritická sekce je část programu, která pracuje se sdíleným prostředkem, například společnou proměnnou, souborem nebo databázovým záznamem. Pokud do ní vstoupí více vláken současně bez ochrany, může vzniknout race condition, tedy chyba závislá na načasování.

Mutex zajišťuje vzájemné vyloučení. Semafor může řídit počet procesů, které mají povolen vstup k omezenému zdroji. Monitor kombinuje data, operace a synchronizaci do jednoho vyššího mechanismu. Správná synchronizace musí zabránit porušení dat, ale zároveň nesmí způsobit uváznutí nebo hladovění procesu.

Klasické úlohy

- **Producent-konzument:** producent vkládá položky do bufferu, konzument je odebírá. Řeší se prázdný/plný buffer.
- **Čtenáři a písaři:** více čtenářů může číst současně, písař potřebuje výhradní přístup.
- **Filozofové:** model sdílených vidliček ukazuje deadlock, kdy každý drží jeden zdroj a čeká na další.
- **Deadlock:** vzájemné čekání. **Starvation:** proces se dlouho nedostane ke zdroji.

15. Souborové systémy: vlastnosti, omezení, druhy, žurnálování, kvóty

filesystem FAT NTFS ext journaling quota

Osnova odpovědi

1. Definuj souborový systém jako způsob organizace dat na úložišti.
2. Vysvětli soubor, adresář, metadata, oprávnění a cestu.
3. Uveď vlastnosti a omezení: velikost souboru, velikost svazku, názvy, oprávnění, kompatibilita.
4. Popiš běžné systémy FAT32, exFAT, NTFS, ext4, APFS.
5. Vysvětli žurnálování.
6. Vysvětli kvóty a jejich použití.

Výkladové body

Souborový systém určuje, jak jsou soubory pojmenovány, ukládány, vyhledávány a chráněny. Data se ukládají do bloků nebo clusterů a metadata popisují velikost, čas vytvoření, vlastníka, oprávnění a umístění dat. Adresáře vytvářejí hierarchickou strukturu.

FAT32 je jednoduchý a kompatibilní, ale má omezení velikosti souboru přibližně 4 GB. exFAT se hodí pro flash média a větší soubory. NTFS podporuje oprávnění, žurnálování, kompresi, šifrování a kvóty. ext4 je běžný v Linuxu, APFS v moderním macOS.

Žurnál a kvóty

- **Žurnálování:** systém si před změnou zapíše záměr do žurnálu, takže po pádu lze obnovit konzistentní stav.
- **Kvóty:** omezují množství prostoru nebo počet souborů pro uživatele či skupinu.
- **Fragmentace:** soubor může být rozdělen do více částí, což u HDD zhoršuje výkon.
- **Oprávnění:** čtení, zápis, spuštění; u pokročilých systémů ACL.

16. Řešení problému pomocí algoritmu a řešení založené na neuronových sítích: princip, využití, vlastnosti a implementace

algoritmus AI neuronové sítě data

Osnova odpovědi

1. Definuj algoritmus a jeho vlastnosti.
2. Vysvětli klasické algoritmické řešení: pravidla, kroky, vstupy, výstupy.
3. Popiš neuronovou síť jako model učící se z dat.
4. Porovnej výhody a nevýhody obou přístupů.
5. Uveď typické oblasti použití.
6. Vysvětli základní implementační kroky u obou možností.

Výkladové body

Algoritmus je přesný postup řešení úlohy. Má vstupy, výstupy, konečný počet kroků, jednoznačnost a efektivitu. Hodí se tam, kde známe pravidla: výpočet ceny, řazení, vyhledávání, validace formuláře nebo řízení podle jasně daných podmínek.

Neuronová síť se neučí ručně napsaná pravidla, ale vztahy v trénovacích datech. Skládá se z vrstev a neuronů s váhami. Při trénování se porovnává výstup modelu se správným výsledkem a pomocí optimalizace se váhy upravují. Hodí se pro rozpoznávání obrazu, řeči, textu, predikce nebo klasifikaci, kde je obtížné napsat přesná pravidla.

Srovnání

- **Algoritmus:** vysvětlitelný, přesný, závislý na ručně navržených pravidlech.
- **Neuronová síť:** učí se ze vzorů, potřebuje data, může být méně vysvětlitelná.
- **Implementace algoritmu:** analýza, návrh, pseudokód, program, testy.
- **Implementace AI:** sběr dat, čištění, trénování, validace, testování, nasazení, monitoring.

> Neuronová síť není kouzelné univerzální řešení. Bez kvalitních dat, správné metriky a kontroly výsledků může dávat přesvědčivě vypadající chyby.

17. Běh programu: strojový a zdrojový kód, interpretované, kompilované jazyky a hybridní řešení, přenositelnost mezi architekturami

kompilace interpret bytecode architektura

Osnova odpovědi

1. Rozliš zdrojový kód, strojový kód a spustitelný soubor.
2. Vysvětli překlad kompilátorem.
3. Vysvětli interpretaci programu.
4. Popiš hybridní modely: bytecode, virtuální stroj, JIT.
5. Vysvětli přenositelnost mezi OS a procesorovými architekturami.
6. Uveď příklady jazyků a běhových prostředí.

Výkladové body

Zdrojový kód je text napsaný programátorem. Procesor ale provádí strojový kód odpovídající konkrétní instrukční sadě, například x86-64 nebo ARM. Kompilovaný jazyk se před spuštěním přeloží do strojového kódu. Výhodou bývá výkon a možnost kontroly chyb při překladu, nevýhodou nutnost překladu pro konkrétní platformu.

Interpretovaný jazyk se vykonává pomocí interpretu, který příkazy čte a provádí za běhu. Hybridní řešení používají mezikód, například bytecode, který spouští virtuální stroj. JIT kompilace překládá často používané části programu do strojového kódu během běhu, čímž kombinuje přenositelnost a výkon.

Příklady

- **C/C++:** typicky kompilované do nativního kódu.
- **Python:** interpretovaný s mezikódem a virtuálním strojem.
- **Java:** bytecode pro JVM, často JIT kompilace.
- **C#:** mezikód pro .NET runtime.
- **JavaScript:** běží v prohlížeči nebo Node.js, moderní enginy používají JIT.

18. Vývojové diagramy, UML diagramy a dokumentace v softwarovém vývoji

diagramy UML dokumentace analýza

Osnova odpovědi

1. Vysvětli účel diagramů ve vývoji softwaru.
2. Popiš vývojový diagram a základní značky.
3. Uveď hlavní UML diagramy a kdy se používají.
4. Vysvětli dokumentaci analytickou, technickou, uživatelskou a provozní.
5. Popiš vztah dokumentace k týmové spolupráci a údržbě systému.
6. Zmiň, že dokumentace má být aktuální, přiměřená a srozumitelná.

Výkladové body

Vývojový diagram znázorňuje algoritmus pomocí symbolů. Začátek a konec se kreslí terminátorem, operace obdélníkem, rozhodnutí kosočtvercem, vstup a výstup rovnoběžníkem a tok šipkami. Pomáhá pochopit logiku programu ještě před psaním kódu.

UML je standardizovaný jazyk pro modelování softwarových systémů. Diagram tříd popisuje třídy, atributy, metody a vztahy. Use case diagram ukazuje aktéry a případy užití. Sekvenční diagram popisuje komunikaci objektů v čase. Activity diagram je podobný vývojovému diagramu, ale vhodný pro procesy. State diagram popisuje stavy objektu a přechody mezi nimi.

Dokumentace

- **Analytická:** požadavky, případy užití, procesy.
- **Technická:** architektura, API, databáze, instalace, konfigurace.
- **Uživatelská:** návody pro běžné uživatele.
- **Provozní:** zálohy, monitoring, incidenty, nasazování.
- **V kódu:** komentáře a názvy mají vysvětlovat záměr, ne opisovat samozřejmost.

19. Relační databáze a SQL/MySQL: CRUD operace, integrita dat, constraint, indexy, transakce

SQL CRUD constraints transactions

Osnova odpovědi

1. Definuj relační databázi, tabulku, řádek, sloupec a klíč.
2. Vysvětli SQL a jeho části: DDL, DML, DCL, TCL.
3. Popiš CRUD operace.
4. Vysvětli integritu dat a omezení.
5. Popiš indexy a jejich vliv na výkon.
6. Vysvětli transakce a vlastnosti ACID.

Výkladové body

Relační databáze ukládá data do tabulek propojených klíči. Primární klíč jednoznačně identifikuje řádek, cizí klíč odkazuje na řádek v jiné tabulce. SQL je jazyk pro definici struktury, manipulaci s daty, řízení práv a transakcí.

CRUD znamená create, read, update, delete. V SQL tomu odpovídá `INSERT`, `SELECT`, `UPDATE` a `DELETE`. Integritu dat hlídají omezení jako `PRIMARY KEY`, `FOREIGN KEY`, `UNIQUE`, `NOT NULL`, `CHECK` a výchozí hodnoty.

Indexy a transakce

- **Index:** datová struktura urychlující vyhledávání a řazení, často B-tree.
- Index zrychluje čtení, ale zpomaluje zápisy a zabírá místo.
- **Transakce:** skupina operací, která se provede celá, nebo se vrátí zpět.
- **ACID:** atomicita, konzistence, izolace, trvalost.
- `COMMIT` potvrdí změny, `ROLLBACK` je vrátí.

20. Relační databáze a SQL/MySQL: typy relací 1:1, 1:M, M:N a propojení pomocí JOIN

vazby JOIN foreign key normalizace

Osnova odpovědi

1. Zopakuj tabulky, primární a cizí klíč.
2. Vysvětli vazbu 1:1.
3. Vysvětli vazbu 1:M.
4. Vysvětli vazbu M:N a spojovací tabulku.
5. Popiš základní typy JOIN.
6. Uveď příklad dotazu a praktický důvod propojení tabulek.

Výkladové body

Vazba 1:1 znamená, že jednomu záznamu v první tabulce odpovídá nejvýše jeden záznam ve druhé tabulce, například uživatel a jeho rozšířený profil. Vazba 1:M znamená, že jeden záznam má více souvisejících záznamů, například zákazník a objednávky. Cizí klíč je obvykle na straně M.

Vazba M:N znamená, že více záznamů na jedné straně souvisí s více záznamy na druhé straně, například studenti a kurzy. V relační databázi se řeší spojovací tabulkou, která obsahuje cizí klíče na obě tabulky, například `student_kurz(student_id, kurz_id)`.

JOIN

- **INNER JOIN:** vrátí jen záznamy, které mají odpovídající protějšek v obou tabulkách.
- **LEFT JOIN:** vrátí všechny záznamy z levé tabulky a odpovídající z pravé, jinak NULL.
- **RIGHT JOIN:** obdoba pro pravou tabulku.
- **FULL JOIN:** vrátí odpovídající i neodpovídající záznamy z obou stran, pokud ho DBMS podporuje.
- **CROSS JOIN:** kartézský součin.

> Ukázka: `SELECT z.jmeno, o.id FROM zakaznik z JOIN objednavka o ON o.zakaznik_id = z.id;`

21. Verzovací systém Git: větev, commit, staging area, lokální a vzdálený repozitář, práce s větvemi

Git commit branch remote

Osnova odpovědi

1. Definuj verzovací systém a proč se používá.
2. Vysvětli repozitář, pracovní adresář, staging area a commit.
3. Popiš lokální a vzdálený repozitář.
4. Vysvětli větev a typický workflow.
5. Uveď práci s větvemi: vytvoření, přepnutí, sloučení, řešení konfliktů.
6. Zmiň příkazy a dobrou praxi.

Výkladové body

Git ukládá historii změn projektu. Commit je snímek změn s identifikátorem, autorem, časem a zprávou. Pracovní adresář obsahuje aktuální soubory. Staging area je přípravná oblast, kam vybíráme změny pro další commit. Díky tomu lze do commitu zařadit jen související úpravy.

Lokální repozitář je na počítači vývojáře, vzdálený repozitář bývá na serveru, například GitHub, GitLab nebo interní server. Příkazy `clone`, `fetch`, `pull` a `push` slouží k práci se vzdálenými změnami. Větev je samostatná linie vývoje, používaná třeba pro novou funkci nebo opravu chyby.

Typické příkazy

- `git status` ukáže stav pracovního adresáře.
- `git add` přidá změny do staging area.
- `git commit -m "zprava"` uloží záznam změn.
- `git branch`, `git switch`, `git merge` pracují s větvemi.
- `git pull` stáhne a začlení změny, `git push` odešle commity.

> Konflikt vznikne, když Git neumí automaticky sloučit dvě změny stejné části souboru. Musí ho vyřešit člověk.

22. Skalární datové typy, proměnné, konstanty, reference, desetinná čísla a textové řetězce, operátory a výrazy

datové typy | proměnné | float | string

Osnova odpovědi

1. Definuj datový typ jako určení množiny hodnot a operací.
2. Uveď skalární typy: celé číslo, desetinné číslo, boolean, znak, řetězec.
3. Vysvětli proměnnou, konstantu a inicializaci.
4. Popiš hodnotový typ a referenci podle použitého jazyka.
5. Vysvětli zvláštnosti desetinných čísel a textu.
6. Prober operátory, prioritu a vyhodnocování výrazů.

Výkladové body

Proměnná je pojmenované místo nebo vazba na hodnotu, jejíž obsah se může měnit. Konstanta se po nastavení nemá měnit. Datový typ určuje, jak se hodnota ukládá a jaké operace s ní lze provádět. Boolean uchovává pravda/nepravda, integer celá čísla, floating point desetinná čísla a string text.

Desetinná čísla v počítači často používají binární plovoucí řádovou čárku, například IEEE 754. Některá desetinná čísla nelze uložit přesně, proto může vzniknout chyba zaokrouhlení. Pro peníze se používají celočíselné hodnoty v haléřích nebo decimal typy. Řetězce jsou posloupnosti znaků, dnes typicky v Unicode, často UTF-8.

Operátory

- Aritmetické: `+`, `-`, `*`, `/`, modulo.
- Porovnávací: rovnost, nerovnost, větší, menší.
- Logické: AND, OR, NOT.
- Přiřazovací: přiřazení a kombinované operátory.
- Priorita určuje pořadí vyhodnocení, závorky mají přednost a zvyšují čitelnost.

23. Řídicí a datové struktury: podmínky, přepínače, cykly, pole/seznam, mapa/slovník, vícedimenzionální pole

if switch loops array map

Osnova odpovědi

1. Vysvětli řídicí struktury jako řízení toku programu.
2. Popiš podmínku `if`, `else` a vnořování.
3. Vysvětli přepínač `switch` / `case`.
4. Popiš cykly `for`, `while`, `do while` a průchod kolekcí.
5. Vysvětli datové struktury pole, seznam a mapa/slovník.
6. Popiš vícedimenzionální pole a matici.

Výkladové body

Podmínka umožňuje provést část programu jen při splnění logického výrazu. Přepínač je vhodný pro výběr z více konkrétních hodnot. Cykly opakují blok kódu: `for` se hodí při známém počtu opakování, `while` při opakování podle podmínky, `do while` provede tělo alespoň jednou.

Pole ukládá prvky stejného nebo podobného typu pod indexy. Seznam je dynamická kolekce, která může měnit délku. Mapa, slovník nebo asociativní pole ukládá dvojice klíč-hodnota a umožňuje rychlé vyhledání podle klíče. Vícedimenzionální pole si lze představit jako tabulku, například matice řádků a sloupců.

Dobrá praxe

- Podmínky mají být čitelné; složité výrazy je dobré rozdělit.
- Cykly musí mít jasnou ukončovací podmínku.
- Pozor na chybu o jeden prvek u indexů začínajících nulou.
- Mapa se hodí pro rychlé hledání podle ID, kódu nebo názvu.
- Matice se používají v grafice, matematice, tabulkách nebo dynamickém programování.

24. Výjimky a chyby v programu: konstrukce, zachytávání, vyšší vrstva, vlastní výjimky, logování

exceptions try catch logging errors

Osnova odpovědi

1. Rozliš chyby syntaktické, běhové a logické.
2. Vysvětli výjimku jako mechanismus oznamování výjimečného stavu.
3. Popiš konstrukce `try`, `catch`, `finally` nebo jejich ekvivalenty.
4. Vysvětli zachytávání ve vyšší vrstvě.
5. Popiš vlastní výjimky a kdy je použít.
6. Vysvětli logování výjimek a aktivity uživatele.

Výkladové body

Výjimka vzniká při situaci, kterou běžný tok programu neumí zpracovat, například chybějící soubor, neplatný vstup, výpadek sítě nebo porušení pravidel aplikace. Blok `try` obsahuje rizikový kód, `catch` zachytí konkrétní typ výjimky a `finally` provede úklid, například zavření souboru.

Výjimku není vždy vhodné zachytit hned. Nižší vrstva často neví, jak chybu zobrazit uživateli nebo jak rozhodnout o náhradním postupu. Proto se výjimka může předat výš, kde je více kontextu. Vlastní výjimky pomáhají rozlišit doménové chyby, například nedostatek kreditu nebo neplatný stav objednávky.

Logování

- Log obsahuje čas, úroveň, zprávu, kontext a u výjimky stack trace.
- Úrovně: debug, info, warning, error, critical.
- Nelogovat citlivá data jako hesla, tokeny a celé platební údaje.
- Aktivita uživatele se loguje kvůli auditu, diagnostice a bezpečnosti.
- Chyby pro uživatele mají být srozumitelné, ale nemají prozrazovat interní detaily systému.

25. Funkce: hlavička, parametry, návratový typ, argument, callback, lokální a globální rozsah, vlastnosti dobré funkce

function parameters return scope

Osnova odpovědi

1. Definuj funkci jako pojmenovaný blok kódu se vstupy a výstupem.
2. Popiš hlavičku funkce: název, parametry, návratový typ, případně modifikátory.
3. Rozliš parametr a argument.
4. Vysvětli návratovou hodnotu a vedlejší efekty.
5. Popiš callback jako funkci předanou jiné funkci.
6. Vysvětli lokální a globální rozsah platnosti a vlastnosti dobré funkce.

Výkladové body

Funkce pomáhají rozdělit program na menší části. Parametr je proměnná v definici funkce, argument je konkrétní hodnota předaná při volání. Návratový typ určuje, jakou hodnotu funkce vrací. Některé funkce nevracejí hodnotu, ale provádějí akci, například zápis do souboru nebo výpis na obrazovku.

Callback je funkce předaná jako argument, kterou zavolá jiná funkce později. Používá se při událostech, asynchronním programování, filtrování, řazení nebo obsluze tlačítek. Lokální proměnné existují jen uvnitř funkce, globální proměnné jsou dostupné širší části programu, ale jejich nadměrné používání komplikuje testování a údržbu.

Dobrá funkce

- Má jasný název a jednu hlavní odpovědnost.
- Je přiměřeně krátká a čitelná.
- Má srozumitelné parametry a očekávaný návratový typ.
- Omezuje skryté závislosti na globálním stavu.
- Je testovatelná a rozumně ošetřuje chybové stavy.

26. Objektově orientované programování: objekty a třídy, instance, reference, instanční proměnné a metody třídy, modifikátory přístupu, konstruktor, destruktor

OOP class object constructor

Osnova odpovědi

1. Definuj OOP jako modelování programu pomocí objektů.
2. Vysvětli třídu jako předpis a objekt jako konkrétní instanci.
3. Popiš instanční proměnné a metody.
4. Rozliš instanční a statické členy.
5. Vysvětli reference na objekt.
6. Popiš modifikátory přístupu, konstruktor a destruktor.

Výkladové body

Třída popisuje vlastnosti a chování objektů. Objekt je instance třídy vytvořená za běhu programu. Například třída `Auto` může mít vlastnosti barva, rychlost a SPZ a metody zrychli, brzdi nebo nastartuj. Instanční proměnné patří konkrétnímu objektu, metody popisují jeho chování.

Reference je odkaz na objekt v paměti. Pokud dvě proměnné odkazují na stejný objekt, změna přes jednu se projeví i při přístupu přes druhou. Modifikátory přístupu, například public, private a protected, určují, odkud lze ke členům třídy přistupovat. Konstruktor inicializuje objekt při vytvoření. Destruktor nebo finalizační mechanismus řeší úklid prostředků, podle jazyka často automaticky nebo pomocí správy zdrojů.

Co zdůraznit

- **Objekt:** stav plus chování.
- **Třída:** šablona pro objekty.
- **Instance:** konkrétní objekt vytvořený z třídy.
- **Statický člen:** patří třídě, ne jedné instanci.
- **Zapouzdření:** vnitřní data chráníme a zpřístupňujeme přes metody nebo vlastnosti.

27. Objektově orientované programování: dědičnost a rozhraní, zapouzdření, polymorfismus, abstrakce

inheritance interface polymorphism abstraction

Osnova odpovědi

1. Zopakuj čtyři základní pilíře OOP.
2. Vysvětli zapouzdření jako skrytí interního stavu.
3. Popiš dědičnost jako vztah obecnější a konkrétnější třídy.
4. Vysvětli rozhraní jako smlouvu o metodách.
5. Popiš polymorfismus.
6. Vysvětli abstrakci a rozdíl mezi abstraktní třídou a rozhraním.

Výkladové body

Dědičnost umožňuje vytvořit novou třídu z existující a převzít nebo upravit její vlastnosti a metody. Například `ElektrickeAuto` může dědit z obecné třídy `Auto`. Dědičnost vyjadřuje vztah „je druhem“. Rozhraní naopak říká, jaké operace objekt nabízí, ale nemusí určovat implementaci.

Polymorfismus znamená, že se stejným rozhraním můžeme pracovat s různými konkrétními objekty. Například seznam objektů typu `Platba` může obsahovat platbu kartou, převodem i hotově, a všechny mají metodu `proved()`, která se chová podle konkrétní třídy. Abstrakce vybírá podstatné vlastnosti problému a skrývá nepodstatné detaily.

Dobré rozlišení

- **Zapouzdření:** data nejsou volně přístupná, chrání je rozhraní třídy.
- **Dědičnost:** znovupoužití a specializace, ale při nadužívání vytváří těsnou vazbu.
- **Rozhraní:** umožňuje zaměnitelnost implementací.
- **Polymorfismus:** volání stejné operace vede k různému chování podle objektu.
- **Abstrakce:** práce s pojmy na správné úrovni detailu.

28. REST API: princip fungování, účel, formáty dat, zabezpečení a metody HTTP protokolu

REST HTTP JSON security

Osnova odpovědi

1. Definuj API a REST.
2. Vysvětli zdroje, URL, metody HTTP a stavové kódy.
3. Popiš formáty dat JSON a XML.
4. Uveď princip bezstavovosti.
5. Vysvětli autentizaci a autorizaci.
6. Zmiň HTTPS, tokeny, validaci vstupů, rate limiting a CORS.

Výkladové body

API je rozhraní, přes které spolu komunikují programy. REST API používá principy webu: zdroje jsou identifikované URL a operace se provádějí metodami HTTP. Například `GET /users` získá seznam uživatelů, `POST /users` vytvoří nového, `PUT` nebo `PATCH` upraví a `DELETE` smaže.

REST je typicky bezstavový: server neuchovává konverzační stav klienta mezi požadavky, takže každý požadavek musí nést potřebné informace, například token. Data se často posílají ve formátu JSON, protože je dobře čitelný a snadno zpracovatelný v JavaScriptu i serverových jazycích.

HTTP a bezpečnost

- **Metody:** GET, POST, PUT, PATCH, DELETE, OPTIONS.
- **Kódy:** 200 OK, 201 Created, 400 Bad Request, 401 Unauthorized, 403 Forbidden, 404 Not Found, 500 Server Error.
- **Autentizace:** ověření identity uživatele.
- **Autorizace:** ověření, zda má uživatel právo operaci provést.
- **HTTPS:** šifruje komunikaci a ověřuje server.

29. Architektura klient-server ve webovém prostředí, HTTP protokol, AJAX, uchování relace pomocí cookies

client-server HTTP AJAX cookies

Osnova odpovědi

1. Vysvětli roli klienta a serveru.
2. Popiš životní cyklus HTTP požadavku a odpovědi.
3. Uveď metody, hlavičky, tělo zprávy a stavové kódy.
4. Vysvětli statický a dynamický obsah.
5. Popiš AJAX a asynchronní načítání dat.
6. Vysvětli cookies a uchování relace.

Výkladové body

Ve webové architektuře je klient nejčastěji prohlížeč nebo mobilní aplikace a server poskytuje data, stránky nebo API. HTTP je protokol typu požadavek-odpověď. Klient pošle požadavek s metodou, URL, hlavičkami a případně tělem. Server vrátí stavový kód, hlavičky a tělo odpovědi, například HTML, JSON, obrázek nebo soubor.

AJAX umožňuje načítat data na pozadí bez plného obnovení stránky. Moderně se používá `fetch` nebo knihovny nad ním. Díky tomu může webová aplikace reagovat rychleji, posílat formuláře, načítat seznamy nebo aktualizovat část stránky.

Cookies a relace

- HTTP je samo o sobě bezstavové, proto server nepozná uživatele mezi požadavky bez dodatečného mechanismu.
- Cookie je malý údaj uložený v prohlížeči a posílaný se souvisejícími požadavky.
- Session cookie obvykle obsahuje identifikátor relace, data relace jsou na serveru.
- Bezpečnostní atributy: HttpOnly, Secure, SameSite, expirace, omezení domény a cesty.
- Pozor na krádež relace, CSRF a XSS.

30. Zásady bezpečného chování v online prostoru: PC, mobilní zařízení, prohlížeče, e-mail, sociální sítě a online nakupování

kyberbezpečnost hesla phishing soukromí

Osnova odpovědi

1. Začni principem: chránit identitu, data, zařízení a peníze.
2. Prober bezpečnost osobního počítače a mobilu.
3. Vysvětli aktualizace, antivir, firewall, zálohy a šifrování.
4. Popiš bezpečné chování v prohlížeči a e-mailu.
5. Prober sociální sítě, soukromí a digitální stopu.
6. Uveď zásady online nakupování a plateb.

Výkladové body

Základem je aktualizovaný operační systém, aplikace a prohlížeč, protože aktualizace opravují bezpečnostní chyby. Důležitá jsou silná a jedinečná hesla, ideálně uložená ve správci hesel, a dvoufaktorové ověření. Zařízení by mělo mít zámek obrazovky, šifrování disku a možnost vzdáleného dohledání nebo smazání.

V e-mailu je největší riziko phishing, tedy podvodné zprávy, které se snaží získat heslo, peníze nebo instalovat malware. Je nutné kontrolovat odesílatele, odkazy, přílohy, naléhavý tón a nečekané požadavky. V prohlížeči sleduj HTTPS, neinstaluj podezřelá rozšíření a dávej pozor na falešné stránky napodobující banku nebo e-shop.

Praktická pravidla

- Používat správce hesel a pro každý účet jiné heslo.
- Zapnout 2FA u e-mailu, banky, sociálních sítí a důležitých služeb.
- Pravidelně zálohovat podle pravidla 3-2-1: tři kopie, dvě média, jedna mimo zařízení.
- Na sociálních sítích omezit sdílení osobních údajů, polohy a dokumentů.
- Při nákupu ověřit obchod, recenze, podmínky, zabezpečení platby a podezřele nízké ceny.
- Ve veřejné WiFi nepřistupovat k citlivým účtům bez VPN nebo zabezpečeného spojení.

> U maturity pomůže konkrétní příklad: falešný e-mail od banky, podvodný e-shop, únik hesel nebo ztracený mobil.

Závěrečná poznámka

Tyto podklady jsou pracovní verze pro přípravu. Pro plnohodnotnou patnáctiminutovou odpověď je vhodné u každého tématu doplnit vlastní kreslené schéma, jeden příklad z praxe a krátké opakování klíčových pojmů.