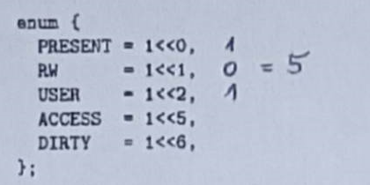
 OSY rychle

* Verze 1.5
  1. Upraveno pár překlepů a odražení
  2. Rozšířeno o termín 25.1.23
* Když už má člověk málo času a potřebuje se to naučit, jsou tu otázky, co se nacházely ve zkouškových testech
* Pro odstranění zvýraznění správné odpovědi doporučuji vše tučně označit a odznačit.
* Snad pomůže, klidně rozšiřujte o další otázky, co se objeví v testech pro budoucí ročníky
* Napsáno za den, takže gramatické chyby se můžou objevit
* Značení

Zeleně – je možnost více správných odpovědí

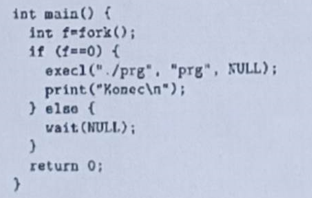
**Tučně** – správná odpově

* Copyleft je:
  1. Myšlenkový směr požadující dostupnost veškerého duševního vlastnictví všem zdarma
  2. **Strategie využívající autorského zákona k šíření uživatelských svobod**
  3. Strategie, jak obejít autorský zákon
* Co musí být nastaveno v tabulce stránek pokud virtuální adresa: 0xAAAAAAAA je součástí kódu uživatelského programu a uživatel z ní může číst, ale nesmí jí modifikovat. Dále požadujeme data z této virtuální adresy byla v paměti RAM na adrese 0x44444AAA



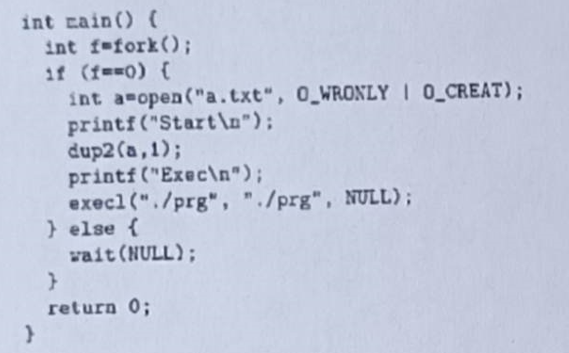
Správné nastavení odpovídající položky v tabulce stránek je:

* 1. **0x44444005**
  2. 0XAAAAA005
  3. 0x11111AAA
  4. 0XAAA44444
  5. 0X44444AAA
* Která následující tvrzení jsou pravdivá?
  1. **Semafor lze vždy použít místo mutexu**
  2. Semafor na rozdíl od mutexu čeká pomocí aktivního čekání ( tzv. busy waiting)
  3. **V moderních OS semafor i mutex obsahuje frontu čekajících vláken**
  4. Mutex lze vždy použít místo semaforu
  5. Mutexy a semafory není možné používat současně v jednom programu
  6. Nic z výše uvedeného není správně
* Mějme následující program



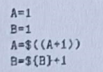
Kdy se vytiskne hlášení “Konec”

* 1. Pokud program “prg” ukončí běh chybou, návratová hodnota programu “prg” bude rozdílná od 0
  2. Vždy po ukončení programu “prg”
  3. **Pokud příkaz execl skončí chybou, neboť nelze spustit program “prg“**
  4. Pokud program “prg“ ukončí běh bez chyby s návratovou hodnotou programu “prg“ 0
* Mějme následující program:



Pokud proběhne program bez chyb, pak:

* 1. **hláška "Start" se vytiskne na standardní vystup, hláška "Exec" bude zapsána do souboru "a.txt“ a program "prg" bude mít přesměrován výstup do souboru "a.txt"**
  2. hláška "Start" a "Exec" se vytiskne na standardní výstup, a program "prg" bude mít přesměrován výstup do souboru "a.txt"
  3. hláška "Start" a "Exec" se vytiskne do souboru "a.txt" a program "prg" bude mít přesměrován výstup do souboru "a.txt"
  4. hláška "Start" se vytiskne na standardní výstup, hláška "Exec" bude zapsána do souboru "a.txt" a program "prg" bude mít výstup přesměrován na standardní výstup rodiče
  5. hláška "Start" a "Exec" se vytiskne na standardní výstup a program 'prg' bude mít vstup také ze standardního vstupu
  6. hláška "Start" se vytiskne na standardní výstup, hláška "Exec" se nevytiskne a program "prg" bude mít výstup do souboru "a.txt"
* Na čem závisí velikost stránek a rámců
  1. **Na architektuře procesoru**
  2. Na velikosti paměti
  3. Na velikosti disku
  4. Na velikosti adresové sběrnice paměti
* Inodový souborový systém
  1. Umožňuje vzdálený přístup do sítě
  2. **Typicky dělí disk na superblok, bitmapy, tabulku inodů a datové bloky**
  3. **Umožňuje spočítat umístění daného inode na disku na základě jeho čísla (indexu)**
  4. Neumožňuje používat tzv. extents, které implementují moderní souborové systémy
  5. **Ukladá informace umístění souborů v datové struktuře zvané inode**
  6. Nic z výše uvedeného není správně
* Systémové volání poll:
  1. **Umožňuje současné čekání na události z více file desriptorů**
  2. Synchronizuje běh více vláken tak, aby bylo možné v jednom okamžiku hlasovat ohledně výsledku paralelně běžící operace
  3. Implementuje smyčku událostí pro grafické aplikace
  4. Slouží k alokaci většího množství paměti (tzv. memory pool)
* Přepínání vláken v rámci jednoho procesu jádro OS řeší pomocí
  1. Vykonání instrukce ctxsw s povolenými přerušeními
  2. **Uložení registrů CPU do kontextu původního vlákna a načtení jejich nové hodnoty z kontextu jiného vlákna**
  3. Změny adresy v registru CR3 tak, aby ukazovala na adresář stránek nového vlákna
  4. Ošetření výpadku stránky (pagefault), který vzniká při přístupu nového vlákna k jeho registrům
  5. Vykonání instrukce ctxsw se zakázanými přerušeními
* Útoku metodou return oriented programming lze zabránit
  1. **Kontrolou integrity zásobníku před návratem z funkce**
  2. Nastavením nespustitelného zásobníku
  3. Randomizací umístění zásobníku
  4. **Randomizací rozložení kódu programu**
  5. Nic z výše uvedeného není správně
* Která tvrzení jsou pravdivá
  1. Program který nepoužívá sdílené prostředky může obsahovat deadlock
  2. Program obsahující deadlock skončí vzájemným čekáním a nikdy nedokončí svojí práci
  3. **Pouze program, který používá sdílené prostředky může obsahovat deadlock**
  4. **Program obsahující deadlock může někdy dokončit svoji práci**
  5. Nic z výše uvedeného není správně
* Jaký bude obsah proměnných A a B po provedení následujícího skriptu:



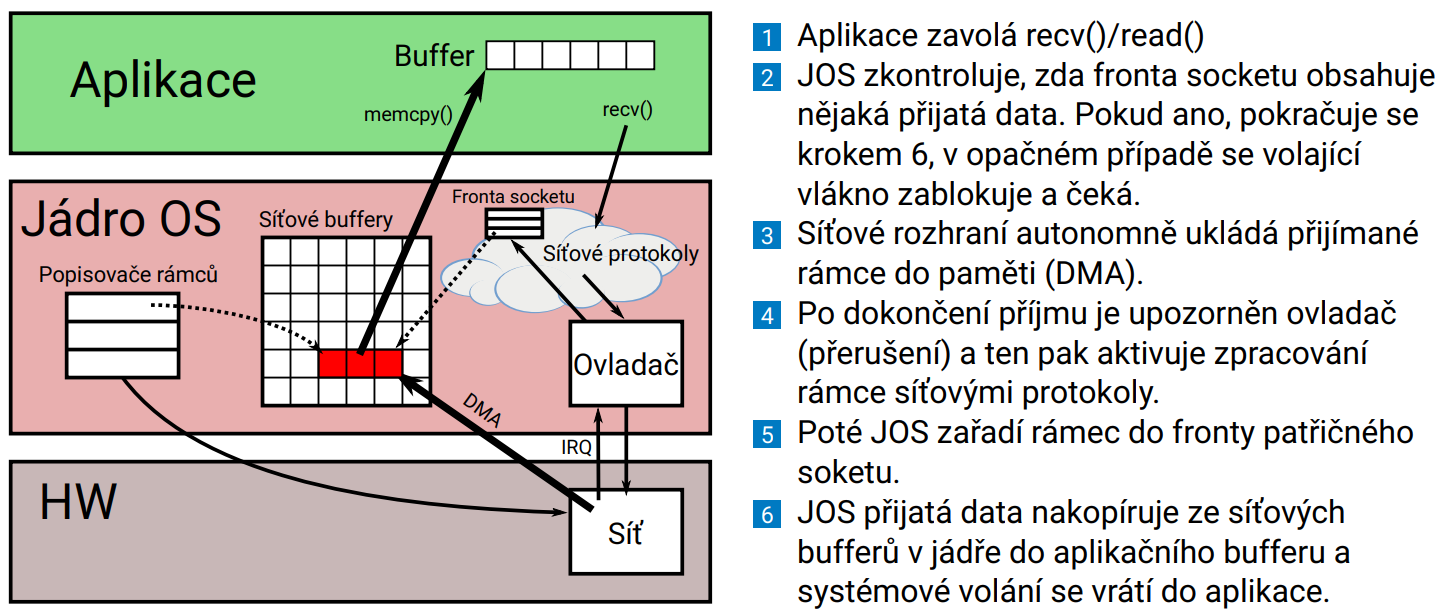
* 1. A je 2, B je 2
  2. A je 1+1, B je 1+1
  3. **A je 2, B je 1+1**
  4. A je 1+1, B je 2
* Hardwarově asistovaná virtualizace
  1. Zrychluje běh virtuálního uživatelského režimu
  2. **Zrychluje vykonávání některých privilegovaných instrukcí**
  3. **Používá mechanismus trap and emulate k emulaci hardware**
  4. Funguje jedině na systému s podporou UEFI firmwarem
  5. **Přidává další vrstvu stránkovacích tabulek**
  6. Nic z výše uvedeného není správně
* Jaká bude hodnota proměnné A po provedení následujícího skriptu:

 X=‘\home\usr\readme.txt‘ A=${X//\\/\/}

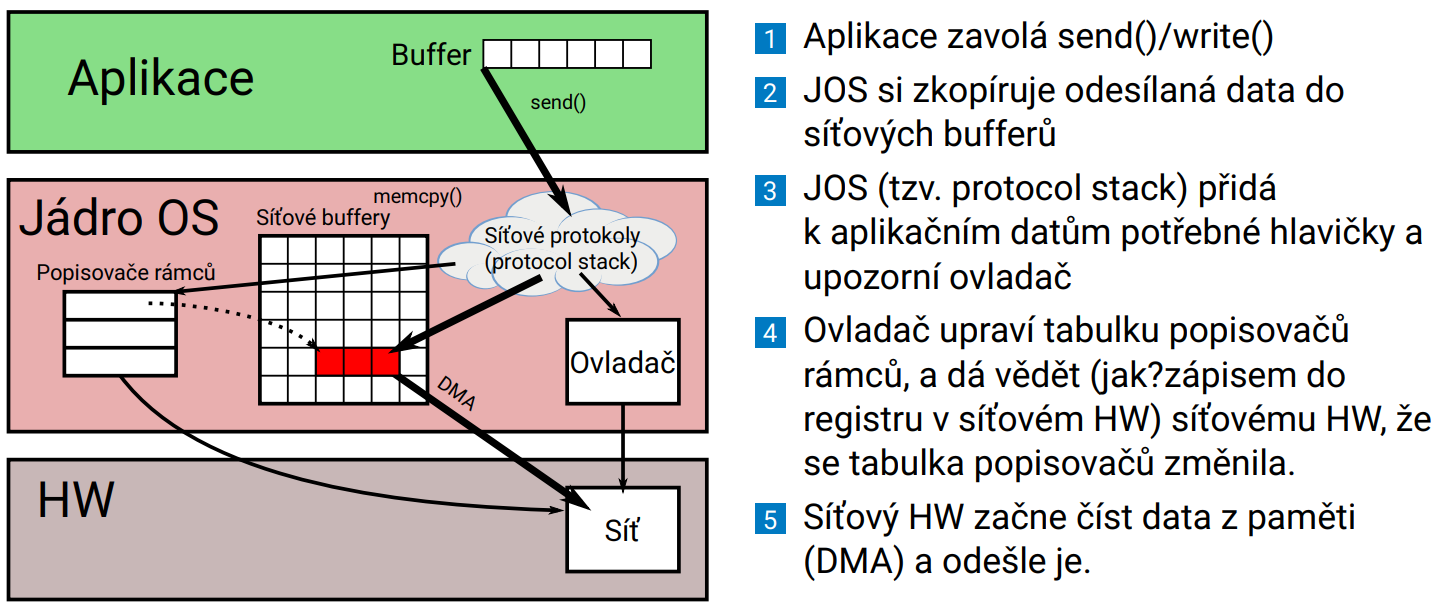
* 1. **/home/usr/readme.txt**
  2. \/home\/usr\/readme.txt
  3. /\/home/\/usr/\/readme.txt
  4. /\home/\usr/\readme.txt
* Aplikace psané v jazyce C/C++ nejčastěji volají služby OS pomocí
  1. **Pomocných funkcí ve standardní knihovně jazyka C (libc)**
  2. Pomocí instrukce call na adresu uvnitř jádra
  3. Vzdáleného volání procedur (RPC)
  4. Služeb systémového procesu systemd (Linux) či svchost.exe (windows)
* Běžná implementace funkce malloc (např. v Glibc):
  1. Nevyužívá paměť uvoněnou dříve funkcí free
  2. Vždy posune konec haldy o velikost požadované paměti
  3. **Se snaží optimálně využít dostupnou paměť**
  4. **Umožňuje alokovat část paměti na haldě (heap)**
  5. Používá se pro definici statických proměnných
  6. Nic z výše uvedeného není správné
* Rozdíl mezi permisivními a recipročními licencemi je:
  1. Reciproční licence chrání uživatele před patentovými hrozbami, kdežto permisivní ne
  2. **Software pod permisivní licencí můžu šířit bez zdrojových kódů, kdežto software pod reciproční licencí ne**
  3. Reciproční licence požadují uvedení původního autora, kdežto permisivní licence ne
* Označte pravdivé výroky: Cyklické plánování – round robin
  1. **Přidělí každému procesu stejné časové kvantum**
  2. Preferuje krátké procesy
  3. Umožňuje rychlé reakce na události v závislosti na předchozí délce běhu programu
  4. **Neumožňuje stárnutí procesů**
  5. Nic z výše uvedeného není správně
* Ukazatel rámce (frame pointer) je:
  1. **Ukazatel na začátek (konec) lokálních proměnných právě vykonávané funkce**
  2. Ukazatel na začátek kódového segmentu aktuálního programu
  3. Ukazatel na začátek stránky paměti, do které uživatelský program právě zapisuje
* UNIXový příkaz strace
  1. Vypisuje, které funkce daný program volá
  2. Vypisuje všechny řetězce tvořené tisknutelnými znaky nalezené v daném souboru
  3. **Vypisuje systémová volání provádění daným programem**
  4. Odstraňuje ze spustitelného souboru ladící informace
* Jak ovladač sdělí hardwaru, že data v paměti může začít zpracovávat (odesílat na síť/zpracovat na disk)?
  1. Zasláním přerušení do HW
  2. **Zápisem do registru daného HW**
  3. Systémovým voláním ioctl (nebo ekvivalentem v neunixových os)
  4. Nijak, HW sám časem zjistí že se v paměti objevila nová data k odeslání
* Souborový systém
  1. Je standard definující název a obsah jednotlivých složek (např. program files, /dev)
  2. Vždy alokuje data jednoho souboru do souvislé oblasti na pevném disku
  3. **Je způsob organizování dat na pevném disku**
  4. Nemůže garantovat konzistenci uložených dat při náhlém vypnutí či pádu počítače
  5. Je typ cloudové služby umožňující ukládaní dat (např. amazon S3
* Kritická sekce je část programu:
  1. Která se musí vykonávat co nejrychleji
  2. **Kde se využívají sdílené zdroje**
  3. Kde dochází ke komunikaci s perifériemi
  4. Kde může dojít k výjimce
* Mechanismus trap-and-emulate se používá
  1. **K virtualizaci vstupu a výstupu (např. síťové rozhraní)**
  2. Jen v systémech bez podpory hardwarové akcelerace virtualizace
  3. **Pokud virtualizovaný systém provede citlivou instrukci**
  4. **K virtualizaci jednotky správy paměti a stránkovacích tabulek**
  5. Nic výše uvedeného není správně
* Jaké způsoby/principy návrhu softwaru vedou ke zvýšení bezpečnosti?
  1. Potvrzování všech prováděných operací uživatelem
  2. Utajení informací o vnitřním logování programu před uživateli
  3. **Co nejjednodušší návrh**
  4. Důsledná kontrola vstupů od potenciálně nedůvěryhodných uživatelů
  5. Nic z výše uvedeného není správně

**Pozor v testu se vyskytuje docela často!** (P9\_S24&S26):

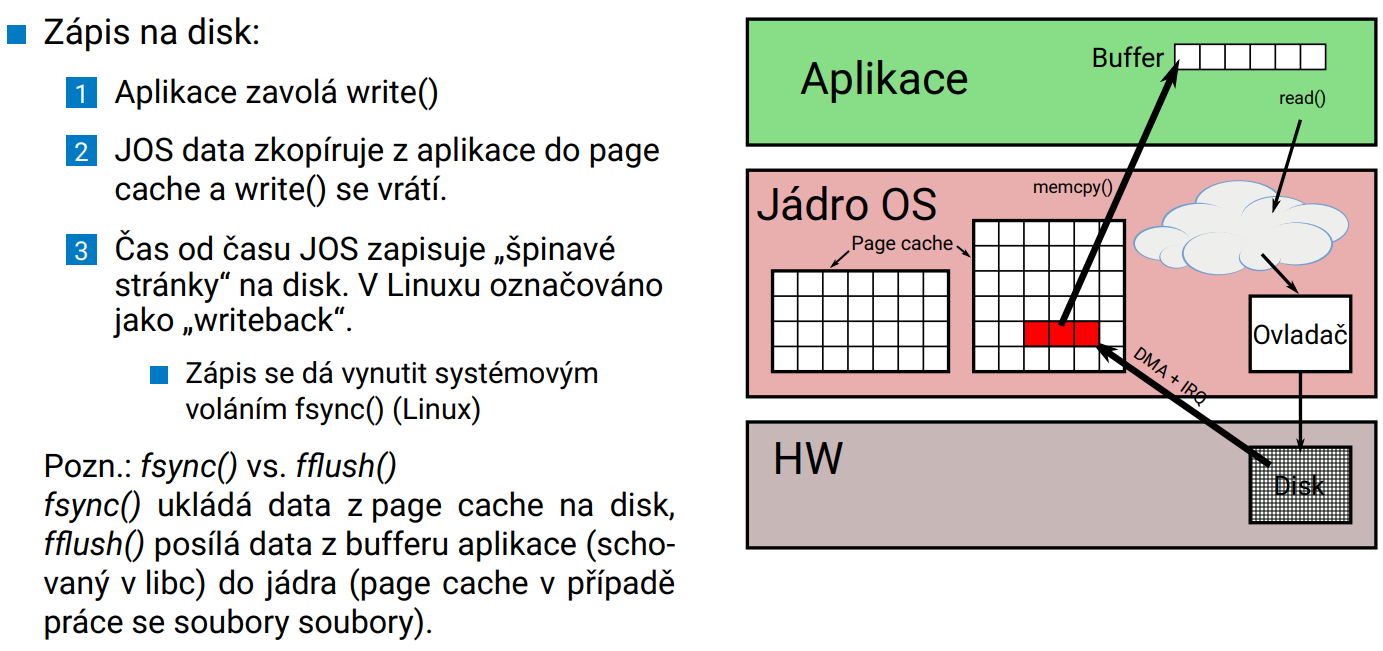
* Popište, jak probíhá příjem dat ze sítě a jaká je jejich cesta skrze jádro OS až do aplikace:



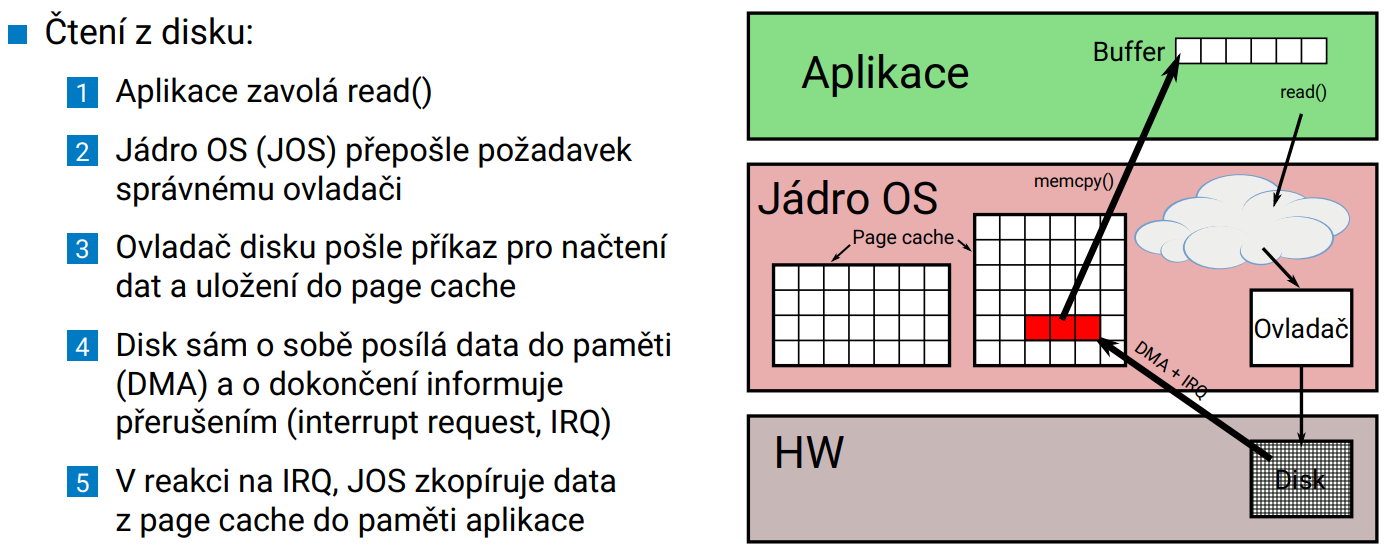
* Popište, jak probíhá odesílání dat do sítě a jaká je jejich cesta skrze jádro OS až do aplikace?



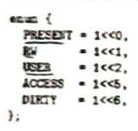
* Jak probíhá ukládání dat z aplikace na pevný disk. Co se s daty děje a jakou roli v tom hraje JOS?



* Jak probíhá načítání dat do aplikace na pevný disk. Co se s daty děje a jakou roli v tom hraje JOS?

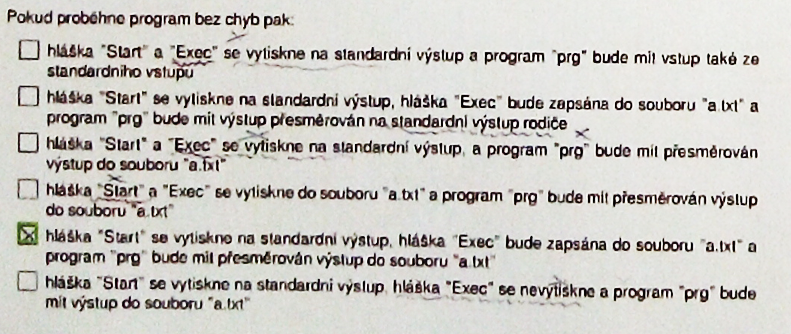
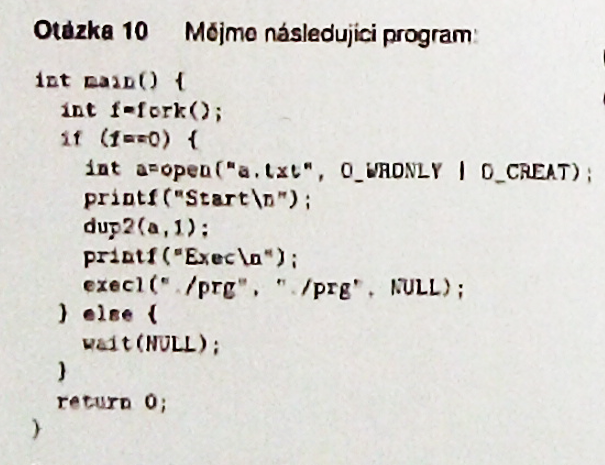
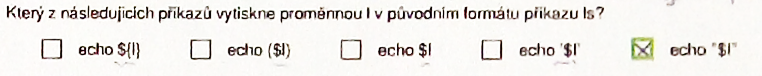


* Co je to Trusted Computing Base?
  1. **Softwarové komponenty kritická pro bezpečnost celého systému**
  2. Základní operace boolovské algebry umožňující OS kontrolu přístupových práv
  3. Pravidla pro návrh bezpečeného softwaru, zejména OS
* Deadlock – vzájené uváznutí – může vzniknout
  1. Pokud jedno vlákno sdílí více společných zdrojů
  2. Pokud alespoň 2 vlákna sdílejí 1 společný zdroj
  3. **Za náhodných okolností, pokud alespoň 2 vlákna sdílejí 2 společné zdroje**
  4. Vždy, pokud vlákna sdílejí 4 společné zdroje
* Žurnálování
  1. **Zajišťuje konzistenci dat (či metadat) na disku pro případ pádu systému**
  2. Zrychluje přístupu k disku pomocí ukládaní dat do vyrovnávací paměti
  3. **Využívá souvislou oblast disku k záznamu diskových transakcí**
  4. Je záznam informací o dění v OS pro účely bezpečnostního auditu
  5. Nic z výše uvedeného není správně
* Které z následujících tvrzení je pravdivé
  1. **Semafor lze použít pro synchronizaci paralelně běžících vláken**
  2. Semafor nemusí pracovat korektně v multiprocesových systémech
  3. Semafor je jediný způsob jak zajistit vstup do kritické sekce
  4. **Semafor řadí čekající vlákna do fronty**
  5. Semafor zaručuje, že nenastane deadlock – vzájemně uváznutí
  6. Nic z výše uvedeného není správně
* Neblokující file desriptor (v unixu s nastaveným příznakem O\_NONBLOCK)
  1. **Způsobuje, že pokud nejsou k dispozici žádná data ke čtení, systémové volání read vrátí chybu**
  2. **Se často používá se systémovými voláními, pool, select či epoll**
  3. Se používá výhradně ve výkonných síťových serverech
  4. Způsobuje, že systémová volání read a write vždy vrátí chybu
  5. Nic z výše uvedeného není správně
* Vlákna mezi sebou sdílejí
  1. **Globální proměnné, dynamicky alokovaná data**
  2. Globální proměnné, dynamicky alokovaná data a lokální proměnné
  3. lokální proměnné a dynamicky alokovaná data
  4. lokální proměnné
* Co musí být nastaveno v tabulce stránek, pokud virtuální adresa: 0x22222222 je součástí kódu uživatelského programu a uživatel z ní může číst i jí modifikovat. Dále požadujeme data z této virtuální adresy byla v paměti RAM na adrese 0x55555222

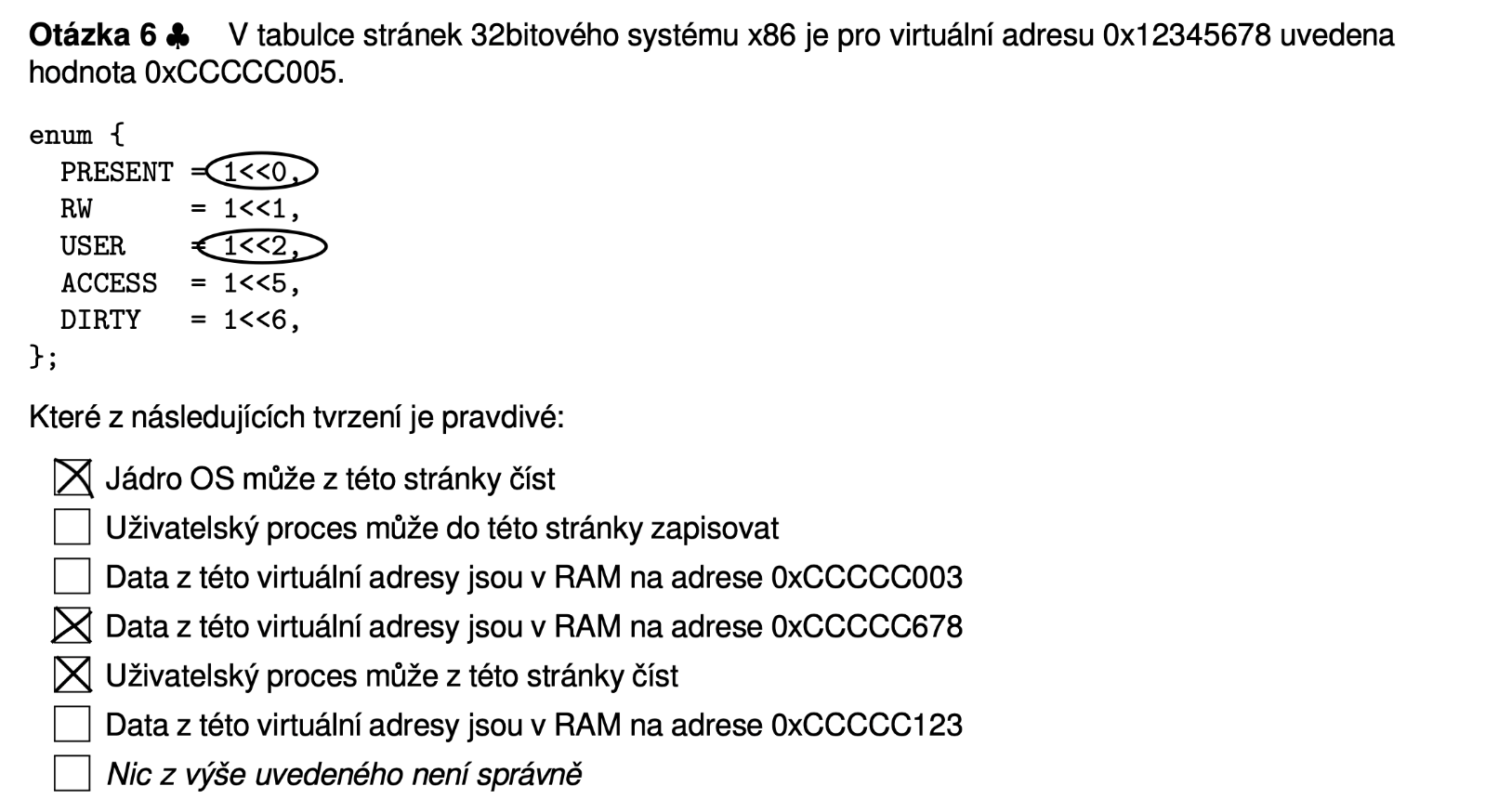


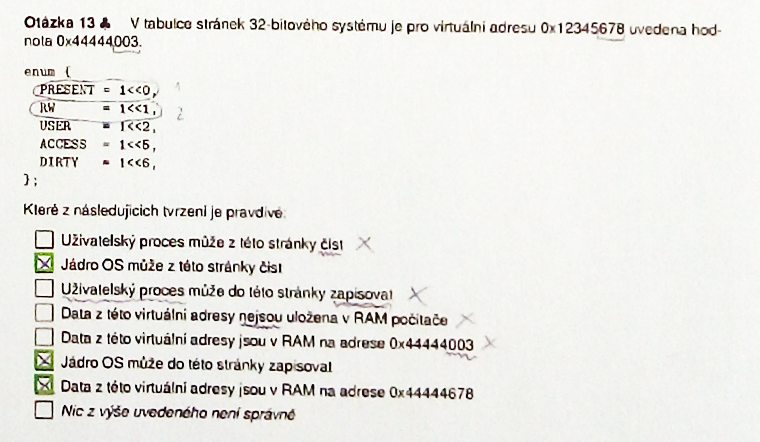
Správné nastavení odpovídající položky v tabulce stránek je:

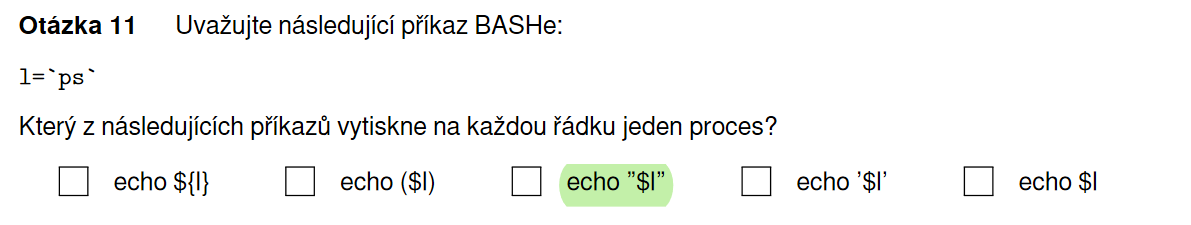
* 1. 0x55555003
  2. 0x55555222
  3. 0x22255555
  4. 0x11111222
  5. **0x55555007**
* Co provádí následující příkaz? tr -d ’/n’
  1. Odstraní znaky n ze standardního vstupu
  2. Zkopíruje standarní vstup beze změny
  3. **Odstraní konce řádek ze standardního vstupu**
  4. Změní všechny znaky d na znak n
* Co provádí následující příkaz? tr -d ’t’
  1. Změní všechny znaky d na znak t
  2. **Odstraní znaky t ze standardního vstupu**
  3. Zkopíruje standardní vstup beze změny
  4. Odstraní tabulátory ze standardního vstupu
* Přístup k periferiím v uživatelském prostoru:
  1. Není nikdy možný
  2. **Je možný s využitím služby jádra**
  3. **Je možný s využitím mapování do paměti procesoru**
  4. Je možný s využitím instrukce test
  5. Nic výše uvedeného není správně
* Takzvané init skripty jsou
  1. Skripty spouštěné po zalogování uživatele, které nastavují například rozložení klávesnice
  2. **Skripty spouštěné některými init systémy pro spouštění a ukončování systémových služeb**
  3. Skripty, které zjednodušují spouštění komplexních aplikací (většinou v jazyce java, např. Eclipse, mathlab apod.)
* Která následující tvrzení jsou pravdivá
  1. **Semafor lze použít pro synchronizaci paralelně běžících vláken**
  2. Při použití semaforu vždy hrozí deadlock
  3. **Semafor lze použít pro správný přístup ke sdíleným datům**
  4. Semafor zaručuje, že nenastane deadlock – vzájemné uváznutí
  5. **Semafor zaručuje spravedlivé čekání na sdílený zdroj**
  6. Nic z výše uvedeného je pravdivé
* Jaké jsou vlastnosti ovladačů zařízení implementovaných v uživatelském prostoru?
  1. Nelze je používat v systémech s monolitickými jádry
  2. Typicky jsou výrazně pomalejší než ovladače v privilegovaném režimu
  3. **Při chybě v ovladači nehrozí pád celého systému**
  4. Při aktualizaci takového ovladače je potřeba restartovat celý systém
* Útoku na systém pomocí přetečení zásobníku se dá zabránit
  1. **Použitím programovacího jazyk, který přetečení automaticky kontroluje**
  2. Odpojením počítače ze sítě
  3. **Randomizaci adresního prostoru (ASLR)**
  4. Použitím antivirového programu
  5. **Zakázáním vykonávání kódu v paměti, kde je zásobník, pomocí atributů stránek**
  6. Nic z výše uvedeného není správně
* Která tvrzení jsou pravdivá?
  1. Deadlocku se není možné vyhnout vhodným plánováním
  2. Deadlock je možné eliminovat, pokud se nebudou používat semafory
  3. **Deadlocku je možné se vyhnout vhodným plánováním, pokud jsou známy požadavky vláken dopředu**
  4. Deadlocku není možné zabránit a může se vyskytnout, kdykoliv se používají sdílené zdroje
  5. **Deadlock se může detekovat, pokud se bude monitorovat čekání na vstup do kritické sekce**
  6. Nic z výše uvedeného není správné



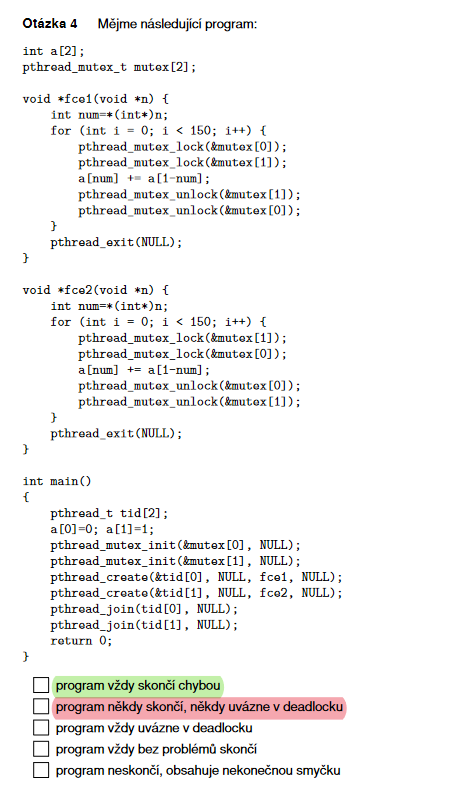
* Licence svobodného software dovolují každému uživateli:
  1. Změnit licence software podle potřeby
  2. **Prodávat software za peníze**
  3. **Software spouštět, studovat, šířit a vylepšovat**
  4. Dělast si se software cokoli bez jakýkoliv omezení
  5. Nic z výše uvedeného není správně
* K zajištění konzistence souborového systému po náhlém vypnutí či pádu systému
  1. Je potřeba používat disková pole, kde jsou data uložena minimálně na dvou místech
  2. **Lze použít metodu, kdy se popis potřebných modifikací nejprve uloží do speciální oblasti na disku**
  3. Je vždy potřeba kompletní kontrola souborového systému po následném zapnutí počítače
  4. Je potřeba vybavit počítač záložním zdrojem (UPS)
  5. Je potřeba provádět všechny související modifikace souborového systému v definovaném pořadí
  6. Nic z výše uvedeného není správně

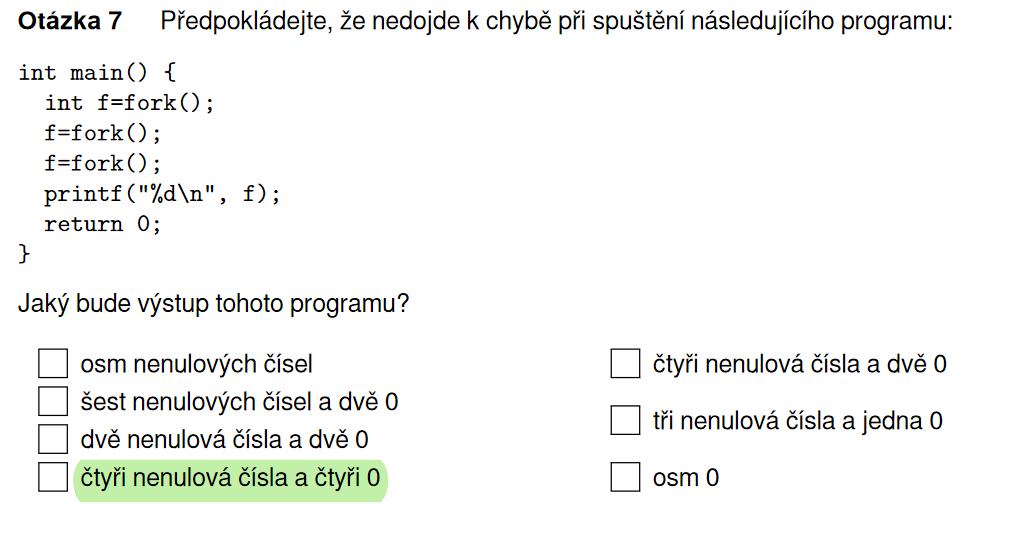




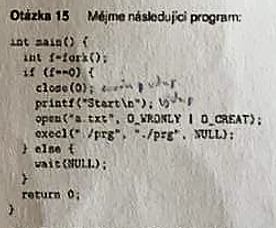


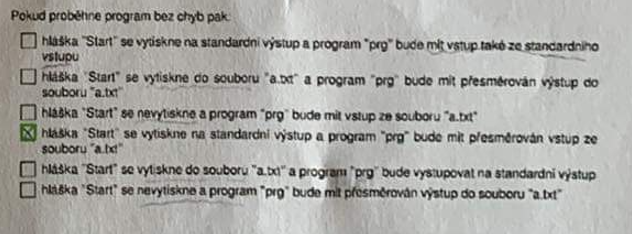
* Co je to výpadek stránky (page fault)
  1. Kritická chyba způsobující havárii jádra OS
  2. **Výjimka CPU, podobně jako například dělení nulou**
  3. Ztráta dat v paměti, programy si musí ztracená data obnovit ze zálohy
  4. **Reakce CPU na situaci, kdy se nějaká instrukce pokouší přistoupit k paměti kde nemá přístup**
  5. Nic z výše uvedeného není správně
* Neblokující I/O
  1. **Dovoluje vláknu dělat něco užitečného, když nejsou k dispozici data, na která čeká**
  2. Garantuje, že se vlákno nikdy nezablokuje a výkon cpu se tak využije na 100%
  3. Je možné používat jen u počítačů se SSD diskem
  4. **Může zabránit dlouhému blokování hlavního vlákna v grafických aplikacích**
  5. Nic z výše uvedeného není správně
* Software šířený pod licencí GPLv2:
  1. není možné prodávat za peníze
  2. nesmí se používat s komerčními OS (např. MS Windows)
  3. **musí být šířen společně se zdrojovými kódy**
  4. je možné použít v proprietárním SW, ale pouze po odstranění všech hlášek ”copyright”
  5. Nic z výše uvedeného není správně
* Která následující tvrzení jsou pravdivá?
  1. Při použití mutexu může vždy nastat deadlock – vzájemné uváznutí
  2. **Mutex lze použít pro správný přístup ke sdíleným datům**
  3. Mutex zaručuje, že nenastane deadlock – vzájemné uváznutí
  4. **Mutex lze použít pro synchronizaci paralelně běžících vláken**
  5. Nic z výše uvedeného není správně
* NEBUDE! Bylo jen pro rok 2018. V roce 2018 zveřejněná zranitelnost CPU zvaná Meltdown:
  1. způsobí, že při vykonání škodlivého kódu se procesor přehřeje a dojde k jeho zničení
  2. je to chyba HW a OS s tím nemůže nic dělat
  3. **umožňuje číst data i ze stránek, ke kterým nemá uživatelský proces přístup**
  4. je jen další mediální bublinou
  5. **vyskytuje se pouze u procesorů od Intelu**
  6. Nic z výše uvedeného není správně
* Stárnutí (starvation) je problémem, který hrozí plánovacím algoritmům. Které algoritmy ohrožuje stárnutí:
  1. cyklické plánování – round robin
  2. **prioritní plánování**
  3. First Come First Served – podle pořadí spuštění
  4. **nejkratší proces první – shortest process next**
  5. Nic z výše uvedeného není správně



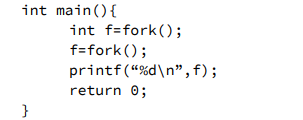


* Hardwarově asistovaná virtualizace (např. VT-x)
  1. zrychluje běh virtuálního uživatelského režimu.
  2. odstraňuje nutnost emulovat hardware (disk, síťová karta, ...) mechanismem trap-and-emulate.
  3. funguje jedině na 32bitovém systému.
  4. zrychluje vykonávání některých privilegovaných instrukcí.
  5. **přidává další vrstvu stránkovacích tabulek.**
  6. Nic z výše uvedeného není správně
* Jaký je vztah pojmů „page fault“ (výpadek stránky) a „segmentation fault“
  1. Segmentation fault běžně nastává během běhu většiny programů
  2. **Page fault běžně nastává během běhu většiny programů**
  3. **Segmentation fault je vždy důsledek page fault**
  4. Dva různé pojmy pro stejnou věc
  5. Page fault je vždy důsledek segmentation fault
  6. **Page fault se používá k implementaci copy-on-write**
  7. Nic z výše uvedeného není správně
* Vlákna na jedno-procesorovém počítači (bez hyper-threadingu):
  1. mohou běžet paralelně pokud nikdy nepoužívají sdílené proměnné
  2. **nemohou běžet paralelně nikdy**
  3. mohou běžet paralelně pouze pokud použijí mutex
  4. mohou běžet paralelně
* Provedení funkce z jádra operačního systému na architektuře x86 lze z uživatelského programu vyvolat:
  1. **instrukcí int s registrem obsahujícím číslo služby jádra**
  2. instrukcí nop s registrem obsahujícím číslo služby jádra
  3. instrukcí call na adresu služby jádra
  4. instrukcí jmp na adresu služby jádra
* Return oriented programming je
  1. Druh útoku na http protokol, který zneužívá hlavičky vrácené serverem
  2. **Způsob, jak spustit kód na napadaném systému i když nemá spustitelný zásobník**
  3. Bezpečný způsob programování bez použití příkazu goto
* Smyčka událostí
  1. **Může pro obsluhu události vytvořit nové vlákno**
  2. Používá se pouze v aplikacích s jedním vláknem
  3. **Je typicky součástí grafických a serverových aplikací**
  4. Se používá pouze v grafických aplikacích psaných ve vyšších programovacích jazycích (java,Csharp)
  5. Slouží v aplikacích typu kalendář k upozorňování uživatelů na blížící se události
  6. Nic z výše uvedeného není správně
* Službu jádra lze na architektuře x86 spustit
  1. Instrukcí out s registrem obsahujícím číslo služby jádra
  2. **Instrukcí int s registrem obsahujícím číslo služby jádra**
  3. Instrukcí call na adresu služby jádra
  4. Instrukcí jmp na adresu služby jádra
* Souborové systémy, které používají tzv. extents
  1. Nejsou vhodné pro nasazení na serverech
  2. **Potřebují méně místa k uložení informací o umístění dat daného souboru**
  3. **Jsou velmi efektivní, pokud data souborů nejsou fragmentovaná**
  4. Potřebují používat nepřímé bloky odkazů na data
  5. Nesmí obsahovat soubory větší než 2GiB
  6. Nic z výše uvedeného není správně
* Která následující tvrzení jsou pravdivá
  1. Při použití semaforu vždy hrozí deadlock
  2. **Semafor lze použít pro zajištění správného přístupu ke sdíleným datům**
  3. **Semafor zaručuje, že všechno čekající vlákna postupně získají sdílený zdroj (=vlákna se nepředbíhají)**
  4. Semafor zaručuje, že nenastane deadlock – vzájemné uváznutí
  5. **Semafor lze použít pro synchronizaci paralelně běžících vláken**
  6. Nic z výše uvedeného je pravdivé





* Uvažujte nasledující príkaz BASHe: P=”HELLO WORLD!”  
  Který příkaz vypíše proměnou P s 3 mezerami mezi slovami HELLO a WORLD!
  1. echo ($P)
  2. echo $(P)
  3. **echo “$P”**
  4. echo ‘$P’
  5. echo $P
* Předpokládejte, že nedojde k chybě při spouštění následujícího programu



Jaký bude výstup tohoto programu?

* 1. jedno nenulové číslo a dvě 0
  2. tři nenulová čísla a jedna 0
  3. jedno nenulové číslo
  4. **dvě nenulová čísla a dvě 0**
* Zneužití chyby přetečení zásobníku se dá zabránit tím, že kompilátor vygeneruje kód, který
  1. Nahradí všechny návraty z funkcí systémového voláním “ret“, které útočník nemůže ovlivnit
  2. Po vstupu do funkce uloží návratovou hodnotu do záložní proměnné
  3. **Vždy před návratem z funkce zkontroluje integritu návratové adresy**
* Seznamy pro řízení přístupu (tzv. ACL)
  1. **Se používají pro určení, zda má být danému uživateli umožněn nebo zamítnut přístup k souboru**
  2. Slouží jako vstupní data při vytváření stránkovacích tabulek nových procesů
  3. Obsahují jména a hesla uživatelů, kteří se smí připojit k danému počítači
* Služby jádra se na architektuře x86 rozlišují
  1. Řetězcem obsahujícím název služby
  2. Adresou, na které je služba umístěna v paměti
  3. Návratovou hodnotou uloženou na zásobníku
  4. **Číslem definujícím typ služby**
* Proč se nesmí volat některé funkce (např. printf) v obslužné rutině signálu (handleru)?
  1. **Protože handler může být volán uprostřed již probíhající funkce printf a její opětovné volání z handleru by pak změnilo hodnoty statických proměnných touto funkcí používaných, což by vedlo k pádu programu.**
  2. Protože funkce jako printf používají signály interně ke komunikaci s jádrem OS a jejich volání v handleru by tak vedlo k nekonečné rekurzi a následnému pádu programu kvůli nedostatku paměti.
  3. Protože handlery signálů jsou vykonávány v kontextu speciálního procesu, který nemá přístup ke standardnímu vstupu a výstupu původního procesu.
* Která tvrzení jsou pravdivá?

1. **Podmínkové proměnné umožňují vláknům čekat na změnu sdílených dat**
2. Čekání na podmínkovou proměnnou lze i bez předchozího zamknutí mutexu
3. Při použití podmínkové proměnné vždy hrozí deadlock
4. **Při čekaní na podmínkovou proměnnou se uvolní propojený mutex**
5. Podmínková proměnná nemá žádný vztah k mutexu
6. Nic z výše uvedeného není správně

* Virtualizace celého systému
  1. Vyžaduje CPU s podporou dvouvrstvého stránkování
  2. **Je možná, jen pokud všechny citlivé instrukce použitého CPU způsobují výjimku**
  3. **Vykonává veškerý kód jádra virtuálního stroje v uživatelském režimu**
  4. Je možná, jen pokud jsou všechny instrukce použitého CPU privilegované
  5. Nic z výše uvedeného není správně

Obsah obrázku text, účtenka

Popis byl vytvořen automaticky¨

* Vyberte pravdivé tvrzení:
  1. Různá vlákna sdílejí registry procesoru
  2. Vlákna sdílejí návratové hodnoty z funkcí.
  3. Vlákna sdílejí parametry funkcí.
  4. **Vlákna sdílejí proměnné deklarované s příznakem static.**
* Jak do proměnné A přiřadíte výstup příkazu ls -l
  1. **A=$(ls -l)**
  2. A=(ls -l)
  3. A=“ls -l“
  4. A=ls -l
  5. A=bash ls -l
* Které z následujících licencí označujeme jako permisivní?
  1. MPL
  2. GPL
  3. **MIT**
  4. **Apache**
  5. LGPL
  6. **BSD**
* Které z následujících licencí označujeme jako reciproční s omezeným působením?
  1. **MPL**
  2. GPL
  3. MIT
  4. Apache
  5. **LGPL**
  6. BSD
* Jak program získá data z klávesnice?

