Ce înțelegem prin overhead spațial și overhead temporal?’

SO 2019-2020: Întrebări examen oral

Aceste întrebări vor fi folosite pentru discuția de la examenul oral de SO. Nu sunt singurele întrebări care vor fi prezente în discuție; de la aceste întrebări discuția va atinge și alte noțiuni prezentate la cursul de SO.

Ce am scris cu verde e opinia mea, nu stiu cat de bune sunt raspunsurile, astept feedback. Am luat de pe wikipedia si din cursuri, also puteti completa si voi :)

# **Stiva software**

Ce este un apel de sistem?

Interfața expusă de sistemul de operare aplicațiilor pentru acțiuni privilegiate se numește interfața de sistem sau interfața de apeluri de sistem sau interfața de servicii de sistem. Un element (o funcționalitate, o funcție, o metodă) din această interfață este numită apel de sistem (system call, syscall) sau serviciu de sistem (system service). Invocarea unui apel de sistem înseamnă schimbarea nivelului de privilegiu de la neprivilegiat (application mode) la privilegiat (supervisor mode). Se spune că un apel de sistem este un trap în modul privilegiat. După apelul de sistem se execută cod de sistem / privileg / kernel.

La revenirea din apel de sistem se trece înapoi în application mode.

TLDR: Un apel de sistem cere privilegii de la kernel pentru a putea executa programul.

De ce sunt necesare apeluri de sistem?

Deoarece asigura o interfata esențiale dintre un proces și sistemul de operare. (Pentru comunicarea proces - OS). Apelurile de sistem sunt singurele puncte de intrare în sistemul de kernel. Toate programele care necesită resurse trebuie să utilizeze apelurile de sistem.

Ce avantaj / dezavantaje au apelurile de sistem?

Avantaj: Simplitate (Ușor de implementat), nu trebuie sa faci operații complexe pentru a deschide/salva un fișier pe disc, în timp ce previn compromiterea OS-ului, driverelor sau a altor componente de sistem.

Dezavantaj: Sunt lente deoarece trebuie setata conexiunea memoriei partajate. Procesele folosesc memoria partajata pentru a crea și atasa apeluri de sistem pentru a obține acces privilegiat (overhead) . Nu sunt portabile (fiecare OS are apelurile sale).

Ce înseamnă user/application mode/space (mod neprivilegiat)? Ce înseamnă kernel/supervisor mode/space (mod privilegiat)?

User mode: Aplicațiile rulează în user mode, nu poate manipula hardware-ul sau toate zonele memoriei. Crash-urile în user mode pot fi rezolvate în mod rezonabil.

Kernel mode: este inaccesibil aplicațiilor, OS-ul expune o interfata (system call interface) prin care aplicațiile solicita acțiuni privilegiate OS-ului. În general componentele de baza ale OS-ului ruleaza in kernel mode.In acest mod orice instrucțiune poate fi executată orice instrucțiune și poate accesa orice zonă de memorie. Crash-urile în kernel mode sunt catastrofale, îți vor distruge pc-ul.

Cum se realizează tranziția în mod privilegiat?

Făcând un apel de sistem.

Ce se întâmplă în momentul tranziției în mod privilegiat? Cum se/Cine asigură (enforcement) existența modului privilegiat?

sistemul de operare are rolul unui furnizor de servicii privilegiate pentru aplicații; o aplicație face o cerere, sistemul de operare verifică dacă aplicația are permisiunile necesare și realizează acțiunea cerută și întoarce rezultatul

sistemul de operare are astfel rol de furnizor de servicii privilegiate, arbitru al folosirii resurselor și monitor al bunei funcționării a sistemului

Ce este o bibliotecă?

O bibliotecă este o colecție de funcții precompilate. În momentul în care un program are nevoie de o funcție, linker-ul va apela respectivă funcție din bibliotecă.

Care este asocierea apel de bibliotecă / apel de sistem?

libc expune apelurile de sistem ca apeluri de bibliotecă

există și apeluri de bibliotecă ce nu sunt asociate cu apel de sistem

apelul de bibliotecă open este construit pentru apelul de sistem open (sau openat)

apelul de bibliotecă fopen este construit peste apelul de sistem open (sau openat)

apelul de bibliotecă strcpy nu are legătură cu un apel de sistem

apelul de bibliotecă malloc face doar uneori apelul de sistem brk; uneori face apelul de sistem mmap (mai multe la capitolul despre memorie)

Care este rolul bibliotecii standard C (libc)?

Implementează funcții C standard și funcții POSIX care necesita apeluri de sistem, facand legatura mai ușor între program și kernel.

Note that not all standard C functions are in libc - most math functions are in libm.

Ce acțiuni se pot executa doar în mod privilegiat?

Întreruperile, instrucțiuni I/O, timer, modificări în memorie si modificari in tabela de statusuri a dispozitivelor.

Ce operații / instrucțiuni low-level (ISA) se pot executa doar în mod privilegiat?

Întreruperile, instrucțiuni I/O, timer, modificări în memorie si modificari in tabela de statusuri a dispozitivelor.

Ce este un sistem de operare monolitic?

performanță mai bună, totul în kernel mode, un apel de sistem face mai multe; dar totul este într-un singur loc, suprafață de atac mare - kernel monolitic

Ce este un sistem de operare de tip microkernel?

sistem de operare mai mic, suprafață de atac mai mică, performantă mai mică: se fac tranziții mai multe user-kernel pentru a permite componentelor software care rulează în user mode să interacționeze - microkernel

Care sunt avantajele unui sistem de operare monolitic?

Performanță mai bună, CPU scheduling, memory/file management, funcții implementate prin syscalls. E un singur proces mare care rulează într-un singur spațiu de adrese.

Care sunt avantajele unui sistem de operare de tip microkernel?

Mai sigur dpdv al securității, dacă are loc un crash, nu crapă tot os-ul ca la monolitic, se poate extinde ușor, design/functionalitate simplista => sistem reliable.

Care tip de sistem de operare are mai multe apeluri de sistem?

Monolitic.

Care este avantajul folosirii mașinilor virtuale din perspectiva securității?

Poți pune mai multe aplicații pe mai multe masini virtuale si atunci cand are loc un atac pe o masina nu va compromite tot sistemul spre deosebire daca am rula acele aplicatii direct pe host. De asemenea, datele pierdute se pot recupera usor.

Ce este o bibliotecă statică? Ce este o bibliotecă dinamică?

Biblioteca statica: set de functii, rutine, variabile care sunt rezolvate la compil  
Biblioteca dinamica: sunt deja incarcate in sistem, la rulare decat se linkeaza

[Static Libraries vs. Dynamic Libraries | by Stuart Kuredjian | Medium](https://medium.com/@StueyGK/static-libraries-vs-dynamic-libraries-af78f0b5f1e4)

Când preferăm folosirea static linking, respectiv dynamic linking? Cu ce diferă un executabil dinamic de un executabil static?

Dați exemplu de apel de sistem blocant.

read(), write(), send(), recv() ,wait()?

Dați exemplu de apel de sistem neblocant.

select(); aio

De ce, în general, o aplicație trebuie să execute un apel de sistem pentru a accesa un dispozitiv hardware? De ce NU poate accesa direct dispozitivul hardware?

Deoarece manipularea hardware-ului este o operatie privilegiata (kernel mode), deci avem nevoie de syscalls pentru a putea trece in kernel mode si pentru a putea accesa dispozitivul. Nu poate accesa direct deoarece aplicatiile ruleaza in user mode unde nu ai privilegii.

Ce înțelegem prin overhead spațial și overhead temporal?

Overhead spatial -> consum mare/ineficient de resurse

Overhead temporal -> timpi mari de executie

Dați exemplu de mecanism / funcție care reduce overhead-ul spațial și unul care reduce overhead-ul temporal.`

Reducerea overhead-ului spatial poate fi atinsa prin alocarea dinamica.

Declararea unei functii inline reduce overheadul temporal deoarece reduce function call overhead, iar apelurile de functii afecteaza overheadul temporal.

Ce înseamnă double buffering? În ce situație concretă (mecanism/funcție) apare?

Se folosesc doua buffere unul pentru citire si unul pentru scriere de exemplu. Se foloseste in functile de I/O din biblioteca stdio.

Sau: ai un buffer in user space si unul in kernel space asta pt I/O

Ce se întâmplă când există o eroare critică (de tip Segmentation fault) la nivelul sistemului de operare?

Programul trimite sistemului de operare eroarea si el incearca sa o rezolve. OS-ul nu stie daca procesul isi poate continua executia si atunci il om0oara. O rezolvare ar fi prin semnale. Pentru a putea rezolva segfaultul, trebuie facut un handle pentru semnalul SIGSEGV.

# 

# **Scheduling**

Ce este un proces?

Un program aflat in executie. Incapsularea unei actiuni in sistemul de calcul: date si cod in memorie, rulare instructiuni in procesor, interactiune cu I/O.

Ce este un thread?

Sunt o varianta simpla a proceselor pentru executarea de actiuni: se mai numesc lightweight processes (LWP).proces

Cu ce diferă un thread de un proces?

Un proces are mai multe threaduri, intrucat un thread este un segment al procesului. Un thread se termina mai repede si se creeaza mai repede.

Cum este afectat spațiul virtual de adrese al unui proces în momentul creării unui thread?

Fiecare thread are propria stiva

Ce zone de memorie au comune thread-urile unui proces și ce zone au specifice?

Zone comune: variabile globale (data/bss), fisierele deschise, spatiul de adresa, modul de tratare al semnalelor.

Zone specifice: registre, stiva, instruction pointer, stare, masca de semnale, thread local storage

Ce conține PCB (Process Control Block)?

Pointeri catre parinte si copil, stadiul, id-ul, prioritate, program counter, registre, pointeri la memorie, lista cu fisiere deschise, limita de memorie

Care sunt stările în care se poate găsi un proces/thread?

RUNNING, WAITING, READY, BLOCKED

Ce efect are apelul fork()?

Creeaza un proces copil ca fiind o copie a procesului parinte.

Ce resurse partajează/nu partajează procesul părinte și procesul copil în cazul apelului fork()?

Partajeaza informatii precum tabela de descriptori de fisiere.

atunci când este creat un proces copil folosind fork(); inițial se partajează \*tot\* spațiul fizic: fiecare dintre procesul părinte și procesul copil are un spațiu virtual propriu, adică o tabelă de pagini proprie; dar spațiile virtuale referă același spațiu fizic, adică tabelele de pagini au același conținut

Ce efect are apelul exec()?

Invoca loaderul si incarca o noua imagine de executabil; “transforma” procesul

Câte procese se pot găsi în starea RUNNING, READY și WAITING?

Unul sau mai multe pentru ready si waiting, dar doar unul pentru running.

La running unul / core.

Ce este o schimbare de context? Ce se întâmplă la o schimbare de context?

Salvarea informatiilor procesului anterior intr-o zona din sistemul de operare si restaurarea informatiilor noului proces. Are loc atunci cand planificatorul decide ca un proces paraseste procesorul.

Ce cauzează schimbări de context?

Atunci cand un proces este scos de pe procesor si este inlocuit de alt proces.

Cauzalitatea se refera fie la schimbare voluntara (e.g. yield, I/O), fie la schimbare involuntara (e.g. expira cuanta de timp)

Ce este o schimbare de context voluntară și o schimbare de context nevoluntară?

Atunci cand un proces decide/cauzeaza schimbarea de context (cedea de bona voie procesorul, executa op blocanta, isi incheie executia). Nevoluntar: planificatorul forteaza procesul (un proces a stat prea mult pe procesor, apare un proces mai important).

Ce sunt thread-urile cu implementare user-level și thread-urile cu implementare kernel-level?

Kernel-level inseamna support la nivelul os-ului, pot folosi procesoarele sistemului(mai multe threadu-ri ale aceluiasi proces pot rula simultan pe un sistem multiprocesor), sunt folosite de planificatorul os-ului, au cuanta de executie, stare de executie si sunt parte din coada ready, schimbarea de context necesita interventia nucleului, e nevoie de suport in kernel.

User-level sunt implementate complet in user-space, nu e nevoie de suport kernel, planificatorul e implementat in user-space, o operatie blocanta blocheaza tot procesul

Thread-urile user level ale unui proces sunt vazute ca un singur proces. => nu se executa paralel

În ce situație este utilă zona TLS (thread local storage)?

Atunci cand vrem sa schimbam valorile unor variabile, dar vrem ca schimbarile sa ramana doar in threadul unde am facut schimbarea si nu global.

De ce este necesară sincronizarea proceselor/thread-urilor?

Pentru a ne asigura ca doua sau mai multe threaduri nu acceseaza in acelasi timp o zona critica.

Ce înseamnă race condition?

Atunci cand doua threaduri modifica aceeasi variabila in acelasi timp. ( acces de citire când datele nu au fost încă actualizate, sau acces după ce datele au fost curățate/refolosite)

Ce înseamnă deadlock?

Un thread intra in starea de asteptare deoarece incearca sa acceseze o resursa care este ocupata de alt thread sau proces, iar daca cel care detine resursa nu o elibereaza atunci se produce deadlock.

Ce înseamnă livelock? Cum diferă de un deadlock?

Atunci cand doua sau mai multe procese acceseaza un lock de mai multe ori si nu se schimba nimic in urma accesarilor. (Un fel de bucla infinita). Similar cu deadlock, diferenta este ca starea proceselor se schimba constant spre deosebire de deadlock unde sunt toate pe waiting.

Care sunt dezavantajele sincronizării?

Scade performanta programului, intrucat threadurile trebuie sa astepte sa astepte pana cand pot accesa o zona critica.

Ce înseamnă TOCTTOU (time of check to time of use)?

O serie de buguri care pot aparea de la race-conditionuri pornind de la verificarea starii unei parti a sistemului si apoi sa folosesti rezultatul acelei verificari, mai exact de la schimbarile care pot aparea intre verificare si utilizarea rezultatului verificarii.

Când se blochează un producător în problema producător-consumator? Dar un consumator?

Un consumator se poate bloca in felul urmator: Consumatorul vede ca bufferul e gol, intra pe if-ul de wait, dar producatorul pune intre timp un item in buffer si apeleaza wakeup inainte sa dea consumatorul sleep => Consumatorul e in deadlock, pentru ca producatorul va produce iteme si consumatorul nu va avea cum sa scoata deoarece el e pe wait. Bufferul se va umple si atunci va da si producatorul sleep si ajungem in situatia cand ambele au deadlock.

Cum se implementează pe un sistem single-core un spinlock? Cum se implementează pe un sistem multi-core un spinlock?

Pentru a atinge sincronizarea, compara valoarea unui pointer cu o anumita valoare si doar daca sunt egale are loc schimbarea

Altă opinie - pentru a putea implementa un spinlock

Cu ce diferă un spinlock de un mutex? Când folosim spinlock-uri? Când folosim mutex-uri?

Spinlockurile continua sa verifice daca locul este liber, mutexul blocheaza direct threadul. Spinlock-ul se foloseste pentru regiuni critice scurte. Mutexul se foloseste pentru regiuni critice mari sau in care threadul se blocheaza.

De la curs: Un mutex conține un spinlock(se implementează pe baza lui)

Ce efect are folosirea operatorului & din shell în crearea unui proces?

Procesul se va rula in background si nu se va astepta incheierea comenzii.

Ce este un proces zombie? Cum apare un proces zombie? Care este problema proceselor zombie?

Proces zombie = proces care si-a incheiat executia dar nu a fost asteptat de parinte.

Orice proces copil trece prin starea de zombie la terminare. <- De ce? dacă procesul copil e așteptat de către părinte nu e tot ok?-> Raspuns:Pentru ca de obicei un proces parinte trebuie sa stie ca un proces copil a fost terminat, sa-i citeasca statusul, de obicei procesele copii stau zombie foarte putin ca sunt curatate de os rapid.

Procesele zombie ocupa memorie in tabela de procese, iar daca ai prea multe procese de genul nu vei mai putea crea procese noi.

Ce este un proces orfan? De ce un proces este orfan foarte puțin timp?

Un proces este orfan daca parintele lui si-a terminat executia. Este orfan pentru scurt timp deoarece este adoptat de procesul init.

Poate fi un proces zombie orfan? Ce se întâmplă cu un proces zombie orfan?

Da, poate fi zombie orfan, daca un parinte moare si nu a asteptat un proces copil zombie, acesta este zombie orfan.

Procesele zombie orfan sunt adoptate de init si se face ripping pe ele(face wait pe ele).

Ce se întâmplă dacă un thread realizează un acces invalid la o zonă de memorie?

Daca nu este implementat un handler de seg fault care doar sa inchida executia pe threadul respectiv, atunci va crapa tot programul.

Poate un thread să acceseze stiva altui thread? Cum?

Da, memoria e partajata în același spațiu virtual de adrese, dacă o funcție într-un thread are un pointer catre adresa stivei altui thread, el va putea sa acceseze stiva acelui thread.

De ce schimbarea de context între două thread-uri ale aceluiași proces este mai rapidă decât schimbarea de context între două thread-uri din procese diferite?

Dacă thread-urile aparțin unor procese diferite e nevoie de schimbarea spațiului de adrese (schimbare de tabelă de pagini, flush la TLB)

Ce forme de comunicare inter-proces cunoști?

Pipes, message queuing, sockets, memorie partajar.

Ce este un semnal? Când se trimite un semnal către un proces?

Un semnal este o întrerupere software, în fluxul normal de execuție a unui proces. Sunt folosite pentru a semnala procesului aparitia unor exceptii, oferindu-i procesului posibilitatea de a actiona.

Cine trimite un semnal unui proces?

Sistemul de operare.

Cum este implementat operatorul | din shell?

Parintele apeleaza pipe() sa obtina o conexiune intre cei doi file descriptori. Copilul scrie intr-unul si celalalt citeste aceleasi date din celalalt fd.

Care sunt avantajele și dezavantajele folosirii memoriei partajate pentru comunicarea inter-proces?

Avantaje: e rapid, procesele pot accesa aceleasi date concurent, modularizare, scalabilitate

Dezavantaje: memorie mai incet de accesat, necesita protectie impotriva accesurilor simultane

Care sunt avantajele și dezavantajele pipe-urilor pentru comunicarea inter-proces?

Avantaje:Pot fi folosite pentru a crea comenzi foarte puternice.

Dezavantaje: Au dimensiune fixa (4MB), in general sunt half-duplex

Ce se întâmplă când toate procesele sistemului sunt blocate?

Idle process

Ce înseamnă waiting time (timp de așteptare) în planificarea proceselor?

Timpul total petrecut de catre proces in modul ready.

Putem avea un sistem multi-core cu un singur proces aflat în starea RUNNING și mai multe procese în READY?

Da

Ce este un proces I/O intensive?

Daca opereaza adesea pe dispozitive I/O.

Ce este un proces CPU intensive?

Daca petrec prea mult timp pe procesor.

Cum tratează planificatorul procesele I/O intensive și procesele CPU intensive?

Procesele I/O intensive au prioritate, pentru că ele stau puțin în starea RUNNING, apoi cedează rândul. Procesele CPU-intensive, pe de altă parte, au o prioritate mai scăzută pentru că folosesc intens resursele(eventual, există riscul de a face sistemul ne-interactiv).

Două thread-uri ale unui proces execută aceeași funcție. Care sunt diferențele între cele două thread-uri?

Ce este un apel thread-safe? Ce este un apel reentrant?

Un apel care poate functiona corect atunci cand se executa simultan de mai multe threaduri

Un apel este reentrant daca poate fi intrerupt si reexecutat fara a afecta flowul programului.

Cum tratăm situația în care apelăm o funcție non-reentrantă într-un handler de semnal?

sigblock

# 

# 

# **Memorie**

Cum asigură sistemul de operare separația între procese?

Memoria virtuala asigura separatia intre procese.

Ce înseamnă mecanismul de memorie virtuală?

O memorie creata de catre OS, care compenseaza lipsa spatiului hard.

Ce reprezintă spațiul virtual de adrese al unui proces?

Reprezinta o multime de adrese virtuale pe care OS-ul le aloca unui proces.

Ce este paginarea memoriei?

Compartimentarea spatiului virtual al fiecarui proces si al spatiului fizic al sistemului in componente de dimensiune fixa, numite pagini (virtuale + fizice(frame))

Ce este fragmentarea internă a memoriei?

Atunci cand se aloca o dimensiune fixa a paginii si ultima pagina nu este ocupata complet, rezultand in fragrmentare interna.

Ce este fragmentarea externă a memoriei?

Memoria este impartita pe segmente (nu este continua).

Ce rol are tabela de pagini?

Mapeaza adresele virtuale la adrese fizice.

Ce este și ce rol are MMU (Memory Management Unit)?

Se ocupa de translatarea proproiu zisa a adreselor virtuale.

Ce rol are TLB?

Retine cele mai recent accesate intrari in tabela de pagini (este un cache), reducand overheaderul.

Ce conține o intrare în tabela de pagini?

Contine adresele virtuale ale paginii si procesul care detine pagina; permisiuni.

frame addr si offset ?(cred)

tabela de pagini e indexată după pagini virtuale: conține adresa paginii fizice corespunzătoare, un bit de validitate, biți de permisiuni

Ce înseamnă tabelă de pagini multi-nivel (ierarhică)? De ce este utilă?

Pentru a avea pagini de dimensiuni dinamice. Tabela multi-nivel este alcatuita din mai multe nivele unde fiecare intrare contine un pointer catre urmatorul nivel.

tabelă de pagini ierarhică: 10 biți (page directory index), 10 biți (PTE index), 12 biți offset

o intrare în page directory referă o structură page table (cu 2^10 intrări), o intrare în tabela de pagini are 2^10 intrări și conține adrese de pagini fizice

Când are loc un TLB miss?

Daca procesorul nu gaseste adresa virtuala in tabela de pagini.

De ce se golește TLB-ul (TLB flush) la schimbare de context?

Se schimba PTBR- e alt proces, deci alt spatiu de adrese virtuale-> alt PT-> alt TLB

De ce nu este nevoie de TLB flush la schimbarea de context între două thread-uri ale aceluiași proces?

Pentru ca nu se modifica VAS-ul. (cred)

Ce înseamnă mecanismul de copy-on-write?

dupa fork zonele sunt read-only, la prima incercare de write se trimite un page fault si se aloca o zona/pagina? noua separata de parinte cu permisiuni de write (cred)

Ce înseamnă demand paging?

rezervarea spatiuliu virtual si maparea lui la o adresa fizica abia cand apare necesitatea, la un prim acces, se aloca memoria physical se adauga in PT(page table) se schimba bitul de validitate si se seteaza permisiunile dupa care se revine la operatia de dinaintea page fault. parca asa era ? (cred)

În ce situație apare page fault fără a cauza segmentation fault?

primul acces al unei zone rezervate si la copy on write (cred)

Ce rol are spațiul de swap?

este folosit cand npteu mai avem spatiu in memorie?

Când are loc swap in și swap out?

swap in cand se aduce in memorie

swap out cand facem swap in si demand pagging

Care este rolul unui page fault. În ce condiții apare?

Bitul valid/invalid din PTE precizează dacă o pagină este validă; dacă nu, rezultă în excepție de acces la memorie (page fault)

Care sunt secțiunile/zonele din spațiul de adrese al unui proces?

text/code , rodata, bss,heap, stack,data ?

Care sunt zonele writable din spațiul de adrese al unui proces?

data, heap?

De ce sunt avantajoase bibliotecile dinamice pentru spațiul de adrese al unui proces?

Două procese sunt pornite din același executabil, ce zone din spațiul de adrese vor partaja?

text si rodata? (cred)

Se alocă un buffer a[100]. De ce a[105] NU va rezulta, în general, în Segmentation fault?

Segmentation fault apare în două situații - când încerc să accesez o zonă de memorie asupra căreia nu am drepturile respective(e.g. am drepturi read-only și vreau să scriu), sau când încerc să accesez o zonă de memorie care nu-mi aparține.  
a[105], deși este în afara vectorului, este foarte probabil să fie în interiorul aceleiași zone de memorie - adică o zonă care îmi aparține și asupra căreia am drepturile corespunzătoare.

În ce situație a[300] rezultă în Segmentation fault?

Deducție din răspunsul precedent -> când a[300] este în afara zonei în care a fost alocat vectorul.

Ce înseamnă maparea unui fișier în memorie? De ce este avantajos să mapăm fișiere față de folosirea read/write?

se foloseste direct bufferul din KS(kernel space) pe cand la read/write se fac syscall adica overhead de timp? (cred) se evită apelurile de sistem read() și write(), deci și doublebuffering-ul.

Câte page fault-uri se pot obține în cazul operației \*a = b?

Ce informații sunt reținute în stivă? Ce variabile C?

Variabile locale argumente de functie si adrese de retur(cred)

În ce zonă sunt reținute variabilele globale inițializate și cele neinițializate?

initializate data si neinitializate bss (cred) -> corect!

Ce înseamnă operația de striping a unui executabil?

**strip** removes symbolic information (and other information not required for execution) from an **executable** file with a view to conserving disk space for production (that is, already debugged) programs.

Ce se întâmplă la faza de loading (încărcarea unui executabil în memorie și crearea unui proces)?

Sunt incarcate datele si codul de catre loader se aloca o stiva procesului si o referinta catre registrul de instrctiuni indicat de header-ul exec. In plus daca e dinamic se linkeaza si bibliotecile externe in spatiul de adrese al procesului(cred)

Ce este entry point-ul într-un executabil?

executabilul are un punct de intrare (entry point) de unde începe execuția noului proces (prin acela se ajunge la funcția main()) (cred)

De ce este necesară prezența unei instrucțiuni de tipul atomic\_compare\_and\_swap în fiecare ISA?

Ce utilitare cunoașteți pentru analiză dinamică și ce utilitare cunoașteți pentru analiză statică?

analiza dinamica - gdb, LLDB, WinDbg, Intel Pin, Valgrind, AddressSanitizer, perf, gcov, Intel VTune

analiza statica - objdump, IDA(iocla), nm, readelf, strings, ldd, radare2, Ghidra

Ce înseamnă analiză statică și ce înseamnă analiză dinamică?

Analiza dinamica se face la rulare.

Analiza statică este analiza pe program fără ca acestea să ruleze. Poate fi analiză statică pe cod sursă sau analiză pe binar / executabil.

Analiza dinamică are loc în momentul rulării programului în proces și are ca țintă principală procesul și resursele folosite de acesta: memorie, registre, fișiere, apeluri de sistem, fluxul de execuție al programului.

Ce înseamnă Stack Guard / Stack Smashing Protection (SSP)?

Se foloseste un canar(canary value) intre stack buffer si adresa de retur. La iesirea din functie daca canary value e modificat(prin buf\_overflow) => segfault.

Ce efect are ASLR (Address Space Layout Randomization)?

randomizeaza adresele zonelor dinamice pentru a preveni atacuri de genul code injection

Ce efect are PIE (Position Independent Executable)?

randomizeaza zonele statice

Ce înseamnă deturnarea fluxului de execuție a unui program (control flow hijack)? De ce este acest lucru relevant pentru un atacator?

Ce înseamnă memory leak / memory disclosure? De ce este acest lucru relevant pentru un atacator?

Alocarea incorecta a memoriei, lasand parti in memorie neeliberate. Memoria kernelului este limitata.

Ce înseamnă că o secvență de cod este PIC (Position Independent Code)?

Cod care o data plasat in memorie ruleaza corect indiferent de adreasa.

De ce în general, preferăm o împărțire a spațiului virtual de adrese între kernel space și user space? Și nu un spațiu dedicat pentru kernel space?

https://stackoverflow.com/questions/5957570/what-is-the-difference-between-the-kernel-space-and-the-user-space

Ce este un code pointer? De ce este interesant din perspectiva securității memoriei?

Ce este un shellcode?

O secventa de cod masina ce poate fi injectata de atacator, se foloseste impreuna cu buffer overflow si poate fi executat doar intr-o zona read-write (cred)

Ce înseamnă code reuse din perspectiva securității memoriei?

Ce înseamnă shell injection din perspectiva securității memoriei?

Ce secvență de cod C va duce la o excepție de acces la memorie (de tip Segmentation fault)? De ce?

Cu ce diferă o funcție de o variabilă într-un executabil și/sau în cadrul spațiului de adrese al unui proces?

Două procese partajează o zonă de memorie. Cum se manifestă acest lucru în tabelele de de pagini ale celor două procese?

Putem avea mai multă memorie fizică decât dimensiunea maximă a spațiului virtual de adrese al unui proces? Dar invers?

# **Fișiere, I/O**

Ce conține un FCB (File Control Block)?

Metadatele unui fișier se găsesc într-o structură numită FCB (File Control Block), (ex. inode pe Unix) • nume (șir de caractere), • identificator (de ex. inode number), • dimensiune, • user (owner), • group, • permisiuni, • timpi de acces Comanda stat în Unix (și apelul stat(2)/fstat(2)) afișează metadatele unui fișier -- numele nu tine de dentry? Ba da.

Ce reprezintă un descriptor de fișier?

Este un număr (întreg) ce referă o intrare în tabela de descriptori de fișier. Este folosit în operații de lucru cu fișiere, pentru a identifica un fișier deschis.

Ce reprezintă tabele de descriptori de fișiere?

Tabela de descriptori de fișier a unui proces conține pointeri; ca structură de date este un vector de pointeri. Acești pointeri referă structuri de fișier deschis de proces. Când un proces deschide un fișier, se alocă o structură de fișier deschis, iar adresa acestei structuri este stocată într-un loc liber (indicat de descriptorul de fișier) din tabela de descriptori de fișier.

Câte tabele de descriptori de fișiere se găsesc într-un sistem de operare?

Fiecare proces are o tabelă de descriptori de fișier, deci vor exista, la nivelul sistemului de operare, atâtea tabele de descriptori de fișier câte procese există în acel moment în sistem.

Ce efect are apelul dup()?

Apelul dup() este folosit practic pentru redirectarea ieșirii, intrării sau erorii standard în fișier. Altă situație practică este pentru operatorul | (pipe) de comunicare între procese. dup() duplica un file descriptor pe prima pozitie available din FDT. Noul descriptor va pointa catre aceeasi structura de fisier deschis ca si descriptorul pe care l-a duplicat.

Ce efect are apelul close()?

Kernel-ul elibereaza file descriptorul asociat.Also resursele asociate structurii open file sunt freed. Structura de fisier deschis are un camp care specifica cati file descriptori pointeaza catre ea. La close se sterge intararea din FDT, se decrementeaza campul respectiv din structura de open file si daca campul devine 0 (nu mai pointeza nimeni catre ea) ea este eliberata.

Ce apeluri modifică pointer-ul/cursorul de fișier (file pointer)?

1. lseek/fseek, apeluri al căror rol este de modificare a cursorului de fișier;

2. read/fread/fgets – la fiecare citire cursorul de fișier este incrementat cu numărul de octeți citiți;

3. write/fwrite/fputs/fprints – la fiecare scriere cursorul de fișier este incrementat cu numărul de octeți scriși;

4. ftruncate – trunchiază fișierul (cursorul este plasat pe 0); - !!!truncate nu schimba cursorul, la fel cum seek nu modifica dimensiunea

5. apelurile echivalente Windows.

si open cred

Ce apeluri modifică dimensiunea fișierului?

Apeluri care pot modifica dimensiunea unui fișier sunt write (poate scrie dincolo de limita unui fișier), ftruncate (modifică chiar câmpul dimensiune) sau open cu argumentulO\_TRUNC care reduce dimensiunea fișierului la 0

Ce efect are apelul/comanda truncate?

Trunchiaza fisierul la o dimensiune specificata. Fisierul poate fi trunchiat la 0 si prin apelul open() cu O\_TRUNC.

Ce este un hard link?

Un hard link se referă la situația în care avem mai multe intrari in directoare (dentry-uri) care pointeaza catre acelasi fișier pe disc (inode). Un nume sau un dentry denotă un hard link.

Ce este un link simbolic/symlink?

Un symlink este un tip de fisier, ii este asociat un inode ca si la celelate tipuri de fisier. In datele sale are un string (cale catre fisierul pe care il refera)

Care este diferența dintre un link simbolic și un hard link?

Un symbolic link are un inode al său, pe când un hard link este un dentry (un nume și un index de inode). Un symbolic link poate referi directoare în timp ce un hard link nu; un symbolic link poate referi un fișier de pe altă partiție/alt sistem de fișiere, în timp ce un hard link nu.

De ce numele unui fișier nu se găsește în inode?

Nu-l gasim deoarece numele sunt stocate in director si fiecare e folosit drept index in tabela de inodes. Mecanismul asta face viata mai usoara atunci cand vrem sa redenumim/mutam un fisier fiindca in felul asta se schimba doar maparea inodurilor in cadrul directoarelor.

Ce se întâmplă în cazul formatării unei partiții?

Se creeaza un nou file system layout in aceea partitie.

Ce se întâmplă cu sistemul de fișiere în cazul folosirii cu succes a comenzii rm?

Folosirea comenzii rm cu success presupune stergerea referintei respectivului obiect din sistemul de fisiere ca o consecinta a unui apel de sistem -> unlink. Se face unlink, se sterge dentry-ul din datele directorului, se decrementeaza nr de linkuri pe care le are inode-ul. Daca nr de link-uri devine 0 inode-ul este sters.

Care este un avantaj al folosirii hard link-urilor și un avantaj al folosirii link-urilor simbolice?

Hard link: putem folosi link-ul si dupa ce fisierul original a fost sters sau mutat.

Symlink: putem sa referim fisiere de pe alte partitii. +directoare

Ce efect are comanda mv /path/to/a.dat /new/path/to/b.dat în sistemul de fișiere?

Se creeaza un nou hard link in noua cale si dupa se face rm (unlink) la primul fisier.

Care sunt tipurile de fișiere pe un sistem de fișiere uzual Unix?

Regular, directory, symbolic link, FIFO special, block special, character special si socket.

Care tipuri de fișiere nu au blocuri de date?

Numai REG, DIR si LNK au date. FIFO, BLK, CHR, SOCKETUNIX nu au date

Ce conțin blocurile de date ale unui director?

Conțin un vector de dentry-uri. Un dentry este o structură ce conține numele fișierului și indexul inode-ului aferent. Fiecare intrare din director (indiferent de tipul acesteia: fișier, director, link symbolic) are un dentry.

Ce este un sistem de fișiere virtual?

As far as Linux is concerned:

Avem mai multe tipuri de sisteme de fisiere. Exemple: ext, ext2...ext4, XFS, JFS, btrfs etc. Fiecare vine cu implementari diferite(Exemplu: modul in care block-urile de fisier sunt alocate sau felul in care directoarele sunt organizate), ceea ce conduce automat la diferente in modul de abordare a lucrului cu fisiere in cadrul unui sistem de operare. Un sistem de fisiere virtual elimina acest inconvenient, prin adaugarea unui layer abstract pentru operatiile sistem-fisier.

Ideile de baza:

- VFS(virtual file sistem) defineste o interfata generica pentru operatiile sistem-fisier. Toate programele care lucreaza cu fisiere folosesc operatiile puse la dispozitie de aceasta interfata generica

- Fiecare sistem de fisiere are cate o implementare pentru interfata VFS

Interfata VFS include operatii corespunzatoare pentru toate apelurile de sistem uzuale pentru lucrul cu sistemele de fisiere / subdirectoare precum: open(), read(), write(), lnameseek(), close(), mmap() etc.

Note: In cadrul WIndows, din VFS lipsesc unele operatii pe care le avem disponibile pe un

sistem de fisiere traditional UNIX. (Microsoft = scumbags)

Ce este un dispozitiv virtual?

Un dispozitiv virtual pointeaza catre un device file care nu are un hardware asociat. In cadrul unui OS precum Linux sau Unix, putem creea un dispozitiv virtual folosind comanda ‘mknod’ si acesta are aceleasi caracteristici precum un dispozitiv real. Kernel-ul il identifica drept un periferic, insa acesta este doar un fisier/director.

Ce tipuri de dispozitive cunoașteți? Clasificați-le din orice punct de vedere cunoașteți

Dpdv UNIX putem distinge doua tipuri de device-uri:

1.Dispozitive caracter(aka serial devices): se ocupa cu gestionarea datelor ‘one character at a time’. Exemple de dispozitive caracter sunt terminalele, tastaturile, mouse-uri, placi de sunet etc. Datele pot fi scrise sau citite o singura data. In situatia in care un byte e citit de la tastatura/mouse, acelasi byte nu poate fi citit de alt program.

2.Dispozitive block(aka random access devices): sunt responsabile cu gestionarea datelor ‘one block at a time’. Dimensiunea unui block este in general un multiplu de 512 exprimat in bytes(poate varia in functie de tipul dispozitivului, in general avem 1024) si avem drept exemplu diverse medii de stocare precum disk-uri sau casete(d-alea cu banda magnetica). Aici datele sunt persistente si exprimate in sectiuni contigue si spre deosebire de dispozitivele de tip caracter, putem citi/scrie datele de mai multe ori.

Exemple de device name-uri:

/dev/hda, /dev/hdb, /dev/mouse, /dev/modem, /dev/zero. /dev/null, /dev/tty(consola)

Note: Toate dispozitivele sunt reprezentate sub forma de fisier(in Unix si Linux).

Cu ce diferă un dispozitiv dCe înțelegem prin overhead spațial și overhead temporal?

SO 2019-2020: Întrebări examen oral

Aceste întrebări vor fi folosite pentru discuția de la examenul oral de SO. Nu sunt singurele întrebări care vor fi prezente în discuție; de la aceste întrebări discuția va atinge și alte noțiuni prezentate la cursul de SO.

Ce am scris cu verde e opinia mea, nu stiu cat de bune sunt raspunsurile, astept feedback. Am luat de pe wikipedia si din cursuri, also puteti completa si voi :)

Stiva software

Ce este un apel de sistem?

Interfața expusă de sistemul de operare aplicațiilor pentru acțiuni privilegiate se numește interfața de sistem sau interfața de apeluri de sistem sau interfața de servicii de sistem. Un element (o funcționalitate, o funcție, o metodă) din această interfață este numită apel de sistem (system call, syscall) sau serviciu de sistem (system service). Invocarea unui apel de sistem înseamnă schimbarea nivelului de privilegiu de la neprivilegiat (application mode) la privilegiat (supervisor mode). Se spune că un apel de sistem este un trap în modul privilegiat. După apelul de sistem se execută cod de sistem / privileg / kernel.

La revenirea din apel de sistem se trece înapoi în application mode.

TLDR: Un apel de sistem cere privilegii de la kernel pentru a putea executa programul.

De ce sunt necesare apeluri de sistem?

Deoarece asigura o interfata esențiale dintre un proces și sistemul de operare. (Pentru comunicarea proces - OS). Apelurile de sistem sunt singurele puncte de intrare în sistemul de kernel. Toate programele care necesită resurse trebuie să utilizeze apelurile de sistem.

Ce avantaj / dezavantaje au apelurile de sistem?

Avantaj: Simplitate (Ușor de implementat), nu trebuie sa faci operații complexe pentru a deschide/salva un fișier pe disc, în timp ce previn compromiterea OS-ului, driverelor sau a altor componente de sistem.

Dezavantaj: Sunt lente deoarece trebuie setata conexiunea memoriei partajate. Procesele folosesc memoria partajata pentru a crea și atasa apeluri de sistem pentru a obține acces privilegiat (overhead) . Nu sunt portabile (fiecare OS are apelurile sale).

Ce înseamnă user/application mode/space (mod neprivilegiat)? Ce înseamnă kernel/supervisor mode/space (mod privilegiat)?

User mode: Aplicațiile rulează în user mode, nu poate manipula hardware-ul sau toate zonele memoriei. Crash-urile în user mode pot fi rezolvate în mod rezonabil.

Kernel mode: este inaccesibil aplicațiilor, OS-ul expune o interfata (system call interface) prin care aplicațiile solicita acțiuni privilegiate OS-ului. În general componentele de baza ale OS-ului ruleaza in kernel mode.In acest mod orice instrucțiune poate fi executată orice instrucțiune și poate accesa orice zonă de memorie. Crash-urile în kernel mode sunt catastrofale, îți vor distruge pc-ul.

Cum se realizează tranziția în mod privilegiat?

Făcând un apel de sistem.

Ce se întâmplă în momentul tranziției în mod privilegiat? Cum se/Cine asigură (enforcement) existența modului privilegiat?

sistemul de operare are rolul unui furnizor de servicii privilegiate pentru aplicații; o aplicație face o cerere, sistemul de operare verifică dacă aplicația are permisiunile necesare și realizează acțiunea cerută și întoarce rezultatul

sistemul de operare are astfel rol de furnizor de servicii privilegiate, arbitru al folosirii resurselor și monitor al bunei funcționării a sistemului

Ce este o bibliotecă?

O bibliotecă este o colecție de funcții precompilate. În momentul în care un program are nevoie de o funcție, linker-ul va apela respectivă funcție din bibliotecă.

Care este asocierea apel de bibliotecă / apel de sistem?

libc expune apelurile de sistem ca apeluri de bibliotecă

există și apeluri de bibliotecă ce nu sunt asociate cu apel de sistem

apelul de bibliotecă open este construit pentru apelul de sistem open (sau openat)

apelul de bibliotecă fopen este construit peste apelul de sistem open (sau openat)

apelul de bibliotecă strcpy nu are legătură cu un apel de sistem

apelul de bibliotecă malloc face doar uneori apelul de sistem brk; uneori face apelul de sistem mmap (mai multe la capitolul despre memorie)

Care este rolul bibliotecii standard C (libc)?

Implementează funcții C standard și funcții POSIX care necesita apeluri de sistem, facand legatura mai ușor între program și kernel.

Note that not all standard C functions are in libc - most math functions are in libm.

Ce acțiuni se pot executa doar în mod privilegiat?

Întreruperile, instrucțiuni I/O, timer, modificări în memorie si modificari in tabela de statusuri a dispozitivelor.

Ce operații / instrucțiuni low-level (ISA) se pot executa doar în mod privilegiat?

Întreruperile, instrucțiuni I/O, timer, modificări în memorie si modificari in tabela de statusuri a dispozitivelor.

Ce este un sistem de operare monolitic?

performanță mai bună, totul în kernel mode, un apel de sistem face mai multe; dar totul este într-un singur loc, suprafață de atac mare - kernel monolitic

Ce este un sistem de operare de tip microkernel?

sistem de operare mai mic, suprafață de atac mai mică, performantă mai mică: se fac tranziții mai multe user-kernel pentru a permite componentelor software care rulează în user mode să interacționeze - microkernel

Care sunt avantajele unui sistem de operare monolitic?

Performanță mai bună, CPU scheduling, memory/file management, funcții implementate prin syscalls. E un singur proces mare care rulează într-un singur spațiu de adrese.

Care sunt avantajele unui sistem de operare de tip microkernel?

Mai sigur dpdv al securității, dacă are loc un crash, nu crapă tot os-ul ca la monolitic, se poate extinde ușor, design/functionalitate simplista => sistem reliable.

Care tip de sistem de operare are mai multe apeluri de sistem?

Monolitic.

Care este avantajul folosirii mașinilor virtuale din perspectiva securității?

Poți pune mai multe aplicații pe mai multe masini virtuale si atunci cand are loc un atac pe o masina nu va compromite tot sistemul spre deosebire daca am rula acele aplicatii direct pe host. De asemenea, datele pierdute se pot recupera usor.

Ce este o bibliotecă statică? Ce este o bibliotecă dinamică?

Biblioteca statica: set de functii, rutine, variabile care sunt rezolvate la compil

Biblioteca dinamica: sunt deja incarcate in sistem, la rulare decat se linkeaza

Static Libraries vs. Dynamic Libraries | by Stuart Kuredjian | Medium

Când preferăm folosirea static linking, respectiv dynamic linking? Cu ce diferă un executabil dinamic de un executabil static?

Dați exemplu de apel de sistem blocant.

read(), write(), send(), recv() ,wait()?

Dați exemplu de apel de sistem neblocant.

select(); aio

De ce, în general, o aplicație trebuie să execute un apel de sistem pentru a accesa un dispozitiv hardware? De ce NU poate accesa direct dispozitivul hardware?

Deoarece manipularea hardware-ului este o operatie privilegiata (kernel mode), deci avem nevoie de syscalls pentru a putea trece in kernel mode si pentru a putea accesa dispozitivul. Nu poate accesa direct deoarece aplicatiile ruleaza in user mode unde nu ai privilegii.

Ce înțelegem prin overhead spațial și overhead temporal?

Overhead spatial -> consum mare/ineficient de resurse

Overhead temporal -> timpi mari de executie

Dați exemplu de mecanism / funcție care reduce overhead-ul spațial și unul care reduce overhead-ul temporal.`

Reducerea overhead-ului spatial poate fi atinsa prin alocarea dinamica.

Declararea unei functii inline reduce overheadul temporal deoarece reduce function call overhead, iar apelurile de functii afecteaza overheadul temporal.

Ce înseamnă double buffering? În ce situație concretă (mecanism/funcție) apare?

Se folosesc doua buffere unul pentru citire si unul pentru scriere de exemplu. Se foloseste in functile de I/O din biblioteca stdio.

Sau: ai un buffer in user space si unul in kernel space asta pt I/O

Ce se întâmplă când există o eroare critică (de tip Segmentation fault) la nivelul sistemului de operare?

Programul trimite sistemului de operare eroarea si el incearca sa o rezolve. OS-ul nu stie daca procesul isi poate continua executia si atunci il om0oara. O rezolvare ar fi prin semnale. Pentru a putea rezolva segfaultul, trebuie facut un handle pentru semnalul SIGSEGV.

Scheduling

Ce este un proces?

Un program aflat in executie. Incapsularea unei actiuni in sistemul de calcul: date si cod in memorie, rulare instructiuni in procesor, interactiune cu I/O.

Ce este un thread?

Sunt o varianta simpla a proceselor pentru executarea de actiuni: se mai numesc lightweight processes (LWP).proces

Cu ce diferă un thread de un proces?

Un proces are mai multe threaduri, intrucat un thread este un segment al procesului. Un thread se termina mai repede si se creeaza mai repede.

Cum este afectat spațiul virtual de adrese al unui proces în momentul creării unui thread?

Fiecare thread are propria stiva

Ce zone de memorie au comune thread-urile unui proces și ce zone au specifice?

Zone comune: variabile globale (data/bss), fisierele deschise, spatiul de adresa, modul de tratare al semnalelor.

Zone specifice: registre, stiva, instruction pointer, stare, masca de semnale, thread local storage

Ce conține PCB (Process Control Block)?

Pointeri catre parinte si copil, stadiul, id-ul, prioritate, program counter, registre, pointeri la memorie, lista cu fisiere deschise, limita de memorie

Care sunt stările în care se poate găsi un proces/thread?

RUNNING, WAITING, READY, BLOCKED

Ce efect are apelul fork()?

Creeaza un proces copil ca fiind o copie a procesului parinte.

Ce resurse partajează/nu partajează procesul părinte și procesul copil în cazul apelului fork()?

Partajeaza informatii precum tabela de descriptori de fisiere.

atunci când este creat un proces copil folosind fork(); inițial se partajează \*tot\* spațiul fizic: fiecare dintre procesul părinte și procesul copil are un spațiu virtual propriu, adică o tabelă de pagini proprie; dar spațiile virtuale referă același spațiu fizic, adică tabelele de pagini au același conținut

Ce efect are apelul exec()?

Invoca loaderul si incarca o noua imagine de executabil; “transforma” procesul

Câte procese se pot găsi în starea RUNNING, READY și WAITING?

Unul sau mai multe pentru ready si waiting, dar doar unul pentru running.

La running unul / core.

Ce este o schimbare de context? Ce se întâmplă la o schimbare de context?

Salvarea informatiilor procesului anterior intr-o zona din sistemul de operare si restaurarea informatiilor noului proces. Are loc atunci cand planificatorul decide ca un proces paraseste procesorul.

Ce cauzează schimbări de context?

Atunci cand un proces este scos de pe procesor si este inlocuit de alt proces.

Cauzalitatea se refera fie la schimbare voluntara (e.g. yield, I/O), fie la schimbare involuntara (e.g. expira cuanta de timp)

Ce este o schimbare de context voluntară și o schimbare de context nevoluntară?

Atunci cand un proces decide/cauzeaza schimbarea de context (cedea de bona voie procesorul, executa op blocanta, isi incheie executia). Nevoluntar: planificatorul forteaza procesul (un proces a stat prea mult pe procesor, apare un proces mai important).

Ce sunt thread-urile cu implementare user-level și thread-urile cu implementare kernel-level?

Kernel-level inseamna support la nivelul os-ului, pot folosi procesoarele sistemului(mai multe threadu-ri ale aceluiasi proces pot rula simultan pe un sistem multiprocesor), sunt folosite de planificatorul os-ului, au cuanta de executie, stare de executie si sunt parte din coada ready, schimbarea de context necesita interventia nucleului, e nevoie de suport in kernel.

User-level sunt implementate complet in user-space, nu e nevoie de suport kernel, planificatorul e implementat in user-space, o operatie blocanta blocheaza tot procesul

Thread-urile user level ale unui proces sunt vazute ca un singur proces. => nu se executa paralel

În ce situație este utilă zona TLS (thread local storage)?

Atunci cand vrem sa schimbam valorile unor variabile, dar vrem ca schimbarile sa ramana doar in threadul unde am facut schimbarea si nu global.

De ce este necesară sincronizarea proceselor/thread-urilor?

Pentru a ne asigura ca doua sau mai multe threaduri nu acceseaza in acelasi timp o zona critica.

Ce înseamnă race condition?

Atunci cand doua threaduri modifica aceeasi variabila in acelasi timp. ( acces de citire când datele nu au fost încă actualizate, sau acces după ce datele au fost curățate/refolosite)

Ce înseamnă deadlock?

Un thread intra in starea de asteptare deoarece incearca sa acceseze o resursa care este ocupata de alt thread sau proces, iar daca cel care detine resursa nu o elibereaza atunci se produce deadlock.

Ce înseamnă livelock? Cum diferă de un deadlock?

Atunci cand doua sau mai multe procese acceseaza un lock de mai multe ori si nu se schimba nimic in urma accesarilor. (Un fel de bucla infinita). Similar cu deadlock, diferenta este ca starea proceselor se schimba constant spre deosebire de deadlock unde sunt toate pe waiting.

Care sunt dezavantajele sincronizării?

Scade performanta programului, intrucat threadurile trebuie sa astepte sa astepte pana cand pot accesa o zona critica.

Ce înseamnă TOCTTOU (time of check to time of use)?

O serie de buguri care pot aparea de la race-conditionuri pornind de la verificarea starii unei parti a sistemului si apoi sa folosesti rezultatul acelei verificari, mai exact de la schimbarile care pot aparea intre verificare si utilizarea rezultatului verificarii.

Când se blochează un producător în problema producător-consumator? Dar un consumator?

Un consumator se poate bloca in felul urmator: Consumatorul vede ca bufferul e gol, intra pe if-ul de wait, dar producatorul pune intre timp un item in buffer si apeleaza wakeup inainte sa dea consumatorul sleep => Consumatorul e in deadlock, pentru ca producatorul va produce iteme si consumatorul nu va avea cum sa scoata deoarece el e pe wait. Bufferul se va umple si atunci va da si producatorul sleep si ajungem in situatia cand ambele au deadlock.

Cum se implementează pe un sistem single-core un spinlock? Cum se implementează pe un sistem multi-core un spinlock?

Pentru a atinge sincronizarea, compara valoarea unui pointer cu o anumita valoare si doar daca sunt egale are loc schimbarea

Altă opinie - pentru a putea implementa un spinlock

Cu ce diferă un spinlock de un mutex? Când folosim spinlock-uri? Când folosim mutex-uri?

Spinlockurile continua sa verifice daca locul este liber, mutexul blocheaza direct threadul. Spinlock-ul se foloseste pentru regiuni critice scurte. Mutexul se foloseste pentru regiuni critice mari sau in care threadul se blocheaza.

De la curs: Un mutex conține un spinlock(se implementează pe baza lui)

Ce efect are folosirea operatorului & din shell în crearea unui proces?

Procesul se va rula in background si nu se va astepta incheierea comenzii.

Ce este un proces zombie? Cum apare un proces zombie? Care este problema proceselor zombie?

Proces zombie = proces care si-a incheiat executia dar nu a fost asteptat de parinte.

Orice proces copil trece prin starea de zombie la terminare. <- De ce? dacă procesul copil e așteptat de către părinte nu e tot ok?-> Raspuns:Pentru ca de obicei un proces parinte trebuie sa stie ca un proces copil a fost terminat, sa-i citeasca statusul, de obicei procesele copii stau zombie foarte putin ca sunt curatate de os rapid.

Procesele zombie ocupa memorie in tabela de procese, iar daca ai prea multe procese de genul nu vei mai putea crea procese noi.

Ce este un proces orfan? De ce un proces este orfan foarte puțin timp?

Un proces este orfan daca parintele lui si-a terminat executia. Este orfan pentru scurt timp deoarece este adoptat de procesul init.

Poate fi un proces zombie orfan? Ce se întâmplă cu un proces zombie orfan?

Da, poate fi zombie orfan, daca un parinte moare si nu a asteptat un proces copil zombie, acesta este zombie orfan.

Procesele zombie orfan sunt adoptate de init si se face ripping pe ele(face wait pe ele).

Ce se întâmplă dacă un thread realizează un acces invalid la o zonă de memorie?

Daca nu este implementat un handler de seg fault care doar sa inchida executia pe threadul respectiv, atunci va crapa tot programul.

Poate un thread să acceseze stiva altui thread? Cum?

Da, memoria e partajata în același spațiu virtual de adrese, dacă o funcție într-un thread are un pointer catre adresa stivei altui thread, el va putea sa acceseze stiva acelui thread.

De ce schimbarea de context între două thread-uri ale aceluiași proces este mai rapidă decât schimbarea de context între două thread-uri din procese diferite?

Dacă thread-urile aparțin unor procese diferite e nevoie de schimbarea spațiului de adrese (schimbare de tabelă de pagini, flush la TLB)

Ce forme de comunicare inter-proces cunoști?

Pipes, message queuing, sockets, memorie partajar.

Ce este un semnal? Când se trimite un semnal către un proces?

Un semnal este o întrerupere software, în fluxul normal de execuție a unui proces. Sunt folosite pentru a semnala procesului aparitia unor exceptii, oferindu-i procesului posibilitatea de a actiona.

Cine trimite un semnal unui proces?

Sistemul de operare.

Cum este implementat operatorul | din shell?

Parintele apeleaza pipe() sa obtina o conexiune intre cei doi file descriptori. Copilul scrie intr-unul si celalalt citeste aceleasi date din celalalt fd.

Care sunt avantajele și dezavantajele folosirii memoriei partajate pentru comunicarea inter-proces?

Avantaje: e rapid, procesele pot accesa aceleasi date concurent, modularizare, scalabilitate

Dezavantaje: memorie mai incet de accesat, necesita protectie impotriva accesurilor simultane

Care sunt avantajele și dezavantajele pipe-urilor pentru comunicarea inter-proces?

Avantaje:Pot fi folosite pentru a crea comenzi foarte puternice.

Dezavantaje: Au dimensiune fixa (4MB), in general sunt half-duplex

Ce se întâmplă când toate procesele sistemului sunt blocate?

Idle process

Ce înseamnă waiting time (timp de așteptare) în planificarea proceselor?

Timpul total petrecut de catre proces in modul ready.

Putem avea un sistem multi-core cu un singur proces aflat în starea RUNNING și mai multe procese în READY?

Da

Ce este un proces I/O intensive?

Daca opereaza adesea pe dispozitive I/O.

Ce este un proces CPU intensive?

Daca petrec prea mult timp pe procesor.

Cum tratează planificatorul procesele I/O intensive și procesele CPU intensive?

Procesele I/O intensive au prioritate, pentru că ele stau puțin în starea RUNNING, apoi cedează rândul. Procesele CPU-intensive, pe de altă parte, au o prioritate mai scăzută pentru că folosesc intens resursele(eventual, există riscul de a face sistemul ne-interactiv).

Două thread-uri ale unui proces execută aceeași funcție. Care sunt diferențele între cele două thread-uri?

Ce este un apel thread-safe? Ce este un apel reentrant?

Un apel care poate functiona corect atunci cand se executa simultan de mai multe threaduri

Un apel este reentrant daca poate fi intrerupt si reexecutat fara a afecta flowul programului.

Cum tratăm situația în care apelăm o funcție non-reentrantă într-un handler de semnal?

Memorie

Cum asigură sistemul de operare separația între procese?

Memoria virtuala asigura separatia intre procese.

Ce înseamnă mecanismul de memorie virtuală?

O memorie creata de catre OS, care compenseaza lipsa spatiului hard.

Ce reprezintă spațiul virtual de adrese al unui proces?

Reprezinta o multime de adrese virtuale pe care OS-ul le aloca unui proces.

Ce este paginarea memoriei?

Compartimentarea spatiului virtual al fiecarui proces si al spatiului fizic al sistemului in componente de dimensiune fixa, numite pagini (virtuale + fizice(frame))

Ce este fragmentarea internă a memoriei?

Atunci cand se aloca o dimensiune fixa a paginii si ultima pagina nu este ocupata complet, rezultand in fragrmentare interna.

Ce este fragmentarea externă a memoriei?

Memoria este impartita pe segmente (nu este continua).

Ce rol are tabela de pagini?

Mapeaza adresele virtuale la adrese fizice.

Ce este și ce rol are MMU (Memory Management Unit)?

Se ocupa de translatarea proproiu zisa a adreselor virtuale.

Ce rol are TLB?

Retine cele mai recent accesate intrari in tabela de pagini (este un cache), reducand overheaderul.

Ce conține o intrare în tabela de pagini?

Contine adresele virtuale ale paginii si procesul care detine pagina; permisiuni.

frame addr si offset ?(cred)

tabela de pagini e indexată după pagini virtuale: conține adresa paginii fizice corespunzătoare, un bit de validitate, biți de permisiuni

Ce înseamnă tabelă de pagini multi-nivel (ierarhică)? De ce este utilă?

Pentru a avea pagini de dimensiuni dinamice. Tabela multi-nivel este alcatuita din mai multe nivele unde fiecare intrare contine un pointer catre urmatorul nivel.

tabelă de pagini ierarhică: 10 biți (page directory index), 10 biți (PTE index), 12 biți offset

o intrare în page directory referă o structură page table (cu 2^10 intrări), o intrare în tabela de pagini are 2^10 intrări și conține adrese de pagini fizice

Când are loc un TLB miss?

Daca procesorul nu gaseste adresa virtuala in tabela de pagini.

De ce se golește TLB-ul (TLB flush) la schimbare de context?

Din ratiuni de securitate (cred?).

De ce nu este nevoie de TLB flush la schimbarea de context între două thread-uri ale aceluiași proces?

Pentru ca nu se modifica VAS-ul. (cred)

Ce înseamnă mecanismul de copy-on-write?

dupa fork zonele sunt read-only, la prima incercare de write se trimite un page fault si se aloca o zona/pagina? noua separata de parinte cu permisiuni de write (cred)

Ce înseamnă demand paging?

rezervarea spatiuliu virtual si maparea lui la o adresa fizica abia cand apare necesitatea, la un prim acces, se aloca memoria physical se adauga in PT(page table) se schimba bitul de validitate si se seteaza permisiunile dupa care se revine la operatia de dinaintea page fault. parca asa era ? (cred)

În ce situație apare page fault fără a cauza segmentation fault?

primul acces al unei zone rezervate si la copy on write (cred)

Ce rol are spațiul de swap?

este folosit cand nu mai avem spatiu in memorie?

Când are loc swap in și swap out?

swap in cand se aduce in memorie

swap out cand facem swap in si demand pagging

Care este rolul unui page fault. În ce condiții apare?

Bitul valid/invalid din PTE precizează dacă o pagină este validă; dacă nu, rezultă în excepție de acces la memorie (page fault)

Care sunt secțiunile/zonele din spațiul de adrese al unui proces?

text/code , rodata, bss,heap, stack,data ?

Care sunt zonele writable din spațiul de adrese al unui proces?

data, heap?

De ce sunt avantajoase bibliotecile dinamice pentru spațiul de adrese al unui proces?

Două procese sunt pornite din același executabil, ce zone din spațiul de adrese vor partaja?

text si rodata? (cred)

Se alocă un buffer a[100]. De ce a[105] NU va rezulta, în general, în Segmentation fault?

Segmentation fault apare în două situații - când încerc să accesez o zonă de memorie asupra căreia nu am drepturile respective(e.g. am drepturi read-only și vreau să scriu), sau când încerc să accesez o zonă de memorie care nu-mi aparține.

a[105], deși este în afara vectorului, este foarte probabil să fie în interiorul aceleiași zone de memorie - adică o zonă care îmi aparține și asupra căreia am drepturile corespunzătoare.

În ce situație a[300] rezultă în Segmentation fault?

Deducție din răspunsul precedent -> când a[300] este în afara zonei în care a fost alocat vectorul.

Ce înseamnă maparea unui fișier în memorie? De ce este avantajos să mapăm fișiere față de folosirea read/write?

se foloseste direct bufferul din KS(kernel space) pe cand la read/write se fac syscall adica overhead de timp? (cred) se evită apelurile de sistem read() și write(), deci și doublebuffering-ul.

Câte page fault-uri se pot obține în cazul operației \*a = b?

Ce informații sunt reținute în stivă? Ce variabile C?

Variabile locale argumente de functie si adrese de retur(cred)

În ce zonă sunt reținute variabilele globale inițializate și cele neinițializate?

initializate data si neinitializate bss (cred) -> corect!

Ce înseamnă operația de striping a unui executabil?

strip removes symbolic information (and other information not required for execution) from an executable file with a view to conserving disk space for production (that is, already debugged) programs.

Ce se întâmplă la faza de loading (încărcarea unui executabil în memorie și crearea unui proces)?

Sunt incarcate datele si codul de catre loader se aloca o stiva procesului si o referinta catre registrul de instrctiuni indicat de header-ul exec. In plus daca e dinamic se linkeaza si bibliotecile externe in spatiul de adrese al procesului(cred)

Ce este entry point-ul într-un executabil?

executabilul are un punct de intrare (entry point) de unde începe execuția noului proces (prin acela se ajunge la funcția main()) (cred)

adică adresa primei instrucțiuni ce va fi executată

De ce este necesară prezența unei instrucțiuni de tipul atomic\_compare\_and\_swap în fiecare ISA?

Ce utilitare cunoașteți pentru analiză dinamică și ce utilitare cunoașteți pentru analiză statică?

analiza dinamica - gdb, LLDB, WinDbg.

analiza statica - objdump, IDA(iocla), readelf, nm, strings, ldd, radare2, Ghidra

Ce înseamnă analiză statică și ce înseamnă analiză dinamică?

Analiza dinamica se face la rulare.

Ce înseamnă Stack Guard / Stack Smashing Protection (SSP)?

Se foloseste un canar(canary value) intre stack buffer si adresa de retur. La iesirea din functie daca canary value e modificat(prin buf\_overflow) => segfault.

Ce efect are ASLR (Address Space Layout Randomization)?

randomizeaza adresele zonelor dinamice pentru a preveni atacuri de genul code injection

Ce efect are PIE (Position Independent Executable)?

randomizeaza zonele statice

Ce înseamnă deturnarea fluxului de execuție a unui program (control flow hijack)? De ce este acest lucru relevant pentru un atacator?

Ce înseamnă memory leak / memory disclosure? De ce este acest lucru relevant pentru un atacator?

Alocarea incorecta a memoriei, lasand parti in memorie neeliberate. Memoria kernelului este limitata.

Ce înseamnă că o secvență de cod este PIC (Position Independent Code)?

Cod care o data plasat in memorie ruleaza corect indiferent de adreasa.

De ce în general, preferăm o împărțire a spațiului virtual de adrese între kernel space și user space? Și nu un spațiu dedicat pentru kernel space?

https://stackoverflow.com/questions/5957570/what-is-the-difference-between-the-kernel-space-and-the-user-space

Ce este un code pointer? De ce este interesant din perspectiva securității memoriei?

Ce este un shellcode?

O secventa de cod masina ce poate fi injectata de atacator, se foloseste impreuna cu buffer overflow si poate fi executat doar intr-o zona read-write (cred)

Ce înseamnă code reuse din perspectiva securității memoriei?

Ce înseamnă shell injection din perspectiva securității memoriei?

Ce secvență de cod C va duce la o excepție de acces la memorie (de tip Segmentation fault)? De ce?

Cu ce diferă o funcție de o variabilă într-un executabil și/sau în cadrul spațiului de adrese al unui proces?

Două procese partajează o zonă de memorie. Cum se manifestă acest lucru în tabelele de de pagini ale celor două procese?

Putem avea mai multă memorie fizică decât dimensiunea maximă a spațiului virtual de adrese al unui proces? Dar invers?

Fișiere, I/O

Ce conține un FCB (File Control Block)?

Metadatele unui fișier se găsesc într-o structură numită FCB (File Control Block), (ex. inode pe Unix) • nume (șir de caractere), • identificator (de ex. inode number), • dimensiune, • user (owner), • group, • permisiuni, • timpi de acces Comanda stat în Unix (și apelul stat(2)/fstat(2)) afișează metadatele unui fișier -- numele nu tine de dentry? Ba da.

Ce reprezintă un descriptor de fișier?

Este un număr (întreg) ce referă o intrare în tabela de descriptori de fișier. Este folosit în operații de lucru cu fișiere, pentru a identifica un fișier deschis.

Ce reprezintă tabele de descriptori de fișiere?

Tabela de descriptori de fișier a unui proces conține pointeri; ca structură de date este un vector de pointeri. Acești pointeri referă structuri de fișier deschis de proces. Când un proces deschide un fișier, se alocă o structură de fișier deschis, iar adresa acestei structuri este stocată într-un loc liber (indicat de descriptorul de fișier) din tabela de descriptori de fișier.

Câte tabele de descriptori de fișiere se găsesc într-un sistem de operare?

Fiecare proces are o tabelă de descriptori de fișier, deci vor exista, la nivelul sistemului de operare, atâtea tabele de descriptori de fișier câte procese există în acel moment în sistem.

Ce efect are apelul dup()?

Apelul dup() este folosit practic pentru redirectarea ieșirii, intrării sau erorii standard în fișier. Altă situație practică este pentru operatorul | (pipe) de comunicare între procese. dup() duplica un file descriptor pe prima pozitie available din FDT. Noul descriptor va pointa catre aceeasi structura de fisier deschis ca si descriptorul pe care l-a duplicat.

Ce efect are apelul close()?

Kernel-ul elibereaza file descriptorul asociat.Also resursele asociate structurii open file sunt freed. Structura de fisier deschis are un camp care specifica cati file descriptori pointeaza catre ea. La close se sterge intararea din FDT, se decrementeaza campul respectiv din structura de open file si daca campul devine 0 (nu mai pointeza nimeni catre ea) ea este eliberata.

Ce apeluri modifică pointer-ul/cursorul de fișier (file pointer)?

1. lseek/fseek, apeluri al căror rol este de modificare a cursorului de fișier;

2. read/fread/fgets – la fiecare citire cursorul de fișier este incrementat cu numărul de octeți citiți;

3. write/fwrite/fputs/fprints – la fiecare scriere cursorul de fișier este incrementat cu numărul de octeți scriși;

4. ftruncate – trunchiază fișierul (cursorul este plasat pe 0); - !!!truncate nu schimba cursorul, la fel cum seek nu modifica dimensiunea

5. apelurile echivalente Windows.

si open cred

Ce apeluri modifică dimensiunea fișierului?

Apeluri care pot modifica dimensiunea unui fișier sunt write (poate scrie dincolo de limita unui fișier), ftruncate (modifică chiar câmpul dimensiune) sau open cu argumentulO\_TRUNC care reduce dimensiunea fișierului la 0

Ce efect are apelul/comanda truncate?

Trunchiaza fisierul la o dimensiune specificata. Fisierul poate fi trunchiat la 0 si prin apelul open() cu O\_TRUNC.

Ce este un hard link?

Un hard link se referă la situația în care avem mai multe intrari in directoare (dentry-uri) care pointeaza catre acelasi fișier pe disc (inode). Un nume sau un dentry denotă un hard link.

Ce este un link simbolic/symlink?

Un symlink este un tip de fisier, ii este asociat un inode ca si la celelate tipuri de fisier. In datele sale are un string (cale catre fisierul pe care il refera)

Care este diferența dintre un link simbolic și un hard link?

Un symbolic link are un inode al său, pe când un hard link este un dentry (un nume și un index de inode). Un symbolic link poate referi directoare în timp ce un hard link nu; un symbolic link poate referi un fișier de pe altă partiție/alt sistem de fișiere, în timp ce un hard link nu.

De ce numele unui fișier nu se găsește în inode?

Nu-l gasim deoarece numele sunt stocate in director si fiecare e folosit drept index in tabela de inodes. Mecanismul asta face viata mai usoara atunci cand vrem sa redenumim/mutam un fisier fiindca in felul asta se schimba doar maparea inodurilor in cadrul directoarelor.

Ce se întâmplă în cazul formatării unei partiții?

Se creeaza un nou file system layout in aceea partitie.

Ce se întâmplă cu sistemul de fișiere în cazul folosirii cu succes a comenzii rm?

Folosirea comenzii rm cu success presupune stergerea referintei respectivului obiect din sistemul de fisiere ca o consecinta a unui apel de sistem -> unlink. Se face unlink, se sterge dentry-ul din datele directorului, se decrementeaza nr de linkuri pe care le are inode-ul. Daca nr de link-uri devine 0 inode-ul este sters.

Care este un avantaj al folosirii hard link-urilor și un avantaj al folosirii link-urilor simbolice?

Hard link: putem folosi link-ul si dupa ce fisierul original a fost sters sau mutat.

Symlink: putem sa referim fisiere de pe alte partitii. +directoare

Ce efect are comanda mv /path/to/a.dat /new/path/to/b.dat în sistemul de fișiere?

Se creeaza un nou hard link in noua cale si dupa se face rm (unlink) la primul fisier.

Care sunt tipurile de fișiere pe un sistem de fișiere uzual Unix?

Regular, directory, symbolic link, FIFO special, block special, character special si socket.

Care tipuri de fișiere nu au blocuri de date?

Numai REG, DIR si LNK au date. FIFO, BLK, CHR, SOCKETUNIX nu au date

Ce conțin blocurile de date ale unui director?

Conțin un vector de dentry-uri. Un dentry este o structură ce conține numele fișierului și indexul inode-ului aferent. Fiecare intrare din director (indiferent de tipul acesteia: fișier, director, link symbolic) are un dentry.

Ce este un sistem de fișiere virtual?

As far as Linux is concerned:

Avem mai multe tipuri de sisteme de fisiere. Exemple: ext, ext2...ext4, XFS, JFS, btrfs etc. Fiecare vine cu implementari diferite(Exemplu: modul in care block-urile de fisier sunt alocate sau felul in care directoarele sunt organizate), ceea ce conduce automat la diferente in modul de abordare a lucrului cu fisiere in cadrul unui sistem de operare. Un sistem de fisiere virtual elimina acest inconvenient, prin adaugarea unui layer abstract pentru operatiile sistem-fisier.

Ideile de baza:

- VFS(virtual file sistem) defineste o interfata generica pentru operatiile sistem-fisier. Toate programele care lucreaza cu fisiere folosesc operatiile puse la dispozitie de aceasta interfata generica

- Fiecare sistem de fisiere are cate o implementare pentru interfata VFS

Interfata VFS include operatii corespunzatoare pentru toate apelurile de sistem uzuale pentru lucrul cu sistemele de fisiere / subdirectoare precum: open(), read(), write(), lnameseek(), close(), mmap() etc.

Note: In cadrul WIndows, din VFS lipsesc unele operatii pe care le avem disponibile pe un

sistem de fisiere traditional UNIX. (Microsoft = scumbags)

Ce este un dispozitiv virtual?

Un dispozitiv virtual pointeaza catre un device file care nu are un hardware asociat. In cadrul unui OS precum Linux sau Unix, putem creea un dispozitiv virtual folosind comanda ‘mknod’ si acesta are aceleasi caracteristici precum un dispozitiv real. Kernel-ul il identifica drept un periferic, insa acesta este doar un fisier/director.

Ce tipuri de dispozitive cunoașteți? Clasificați-le din orice punct de vedere cunoașteți

Dpdv UNIX putem distinge doua tipuri de device-uri:

1.Dispozitive caracter(aka serial devices): se ocupa cu gestionarea datelor ‘one character at a time’. Exemple de dispozitive caracter sunt terminalele, tastaturile, mouse-uri, placi de sunet etc. Datele pot fi scrise sau citite o singura data. In situatia in care un byte e citit de la tastatura/mouse, acelasi byte nu poate fi citit de alt program.

2.Dispozitive block(aka random access devices): sunt responsabile cu gestionarea datelor ‘one block at a time’. Dimensiunea unui block este in general un multiplu de 512 exprimat in bytes(poate varia in functie de tipul dispozitivului, in general avem 1024) si avem drept exemplu diverse medii de stocare precum disk-uri sau casete(d-alea cu banda magnetica). Aici datele sunt persistente si exprimate in sectiuni contigue si spre deosebire de dispozitivele de tip caracter, putem citi/scrie datele de mai multe ori.

Exemple de device name-uri:

/dev/hda, /dev/hdb, /dev/mouse, /dev/modem, /dev/zero. /dev/null, /dev/tty(consola)

Note: Toate dispozitivele sunt reprezentate sub forma de fisier(in Unix si Linux).

Cu ce diferă un dispozitiv de tip bloc de un dispozitiv de tip caracter? Dați câte un exemplu de fiecare.

De ce nu are sens operația de seek pe un dispozitiv de tip caracter?

Operatia de seek n-are sens pentru un dispozitiv de tip caracter deoarece tipul asta de device are o singura pozitie, cea curenta(fiindca datele vin sub forma de stream ‘one character at a time’), asadar nu putem sa miscam un pointer in fata sau inapoi in fisier.

‘sudo cat /dev/input/mice’ you’ll see why

Ce adresă IP locală și ce port local are un socket întors de apelul accept()?

int accept(int sockfd, struct sockaddr \*addr, socklen\_t \*addrlen);

Functia returneaza file descriptor-ul socket-ului nou creat, care contine adresa/port-ul asociat peer-ului (aka aplicatia careia ii permitem sa se conecteze)

Ce valoare poate întoarce un apel read() sau un apel write()?

Un apel de sistem read() sau write() intoarce numarul de bytes cititi/scrisi sau 0 in cazul

end-of-file la read().

Ce operații se pot face pe fișiere?

Operatiile care sunt definite in structura de file\_operations din cadrul kernelului, din care se pot aminti cateva mai populare: read(), llseek(), poll(), ioctl(), mmap(), write() etc.

Avem user space buffer si kernel buffer cu rolul de a limita numarul de accesari directe pe fisierele de pe disc.

Ce operații asupra fișierelor modifică/nu modifică valoarea cursorului unui fișier?

Nu modifica cursorul: operatiile de schimbare ale atributelor unui fisier, operatiile de open/close/create.

Modifica cursorul: operatiile de pozitionare intr-un fisier(obviously), scrierea/citirea si trunchierea.

Ce operații asupra fișierelor modifică/nu modifică dimensiunea fișierului?

Modifica: write daca scrii peste dimensiunea curenta, truncate

Nu modifica: lseek, read

Unde este reținută valoarea cursorului de fișiere (file pointer) și unde este reținută dimensiunea fișierului?

Dimensiunea fisierului e retinuta in structura inod-ului(pe langa owner id, group id, flag-uri de sistem/user, device id etc), iar file pointer-ul rezida in cadrul kernel-ului, acesta avand rolul de I/O managing in File Descriptor Table. (le gasesti de obicei la /proc/\_\_PID\_\_/fd)

De ce avem două buffere asociate fiecărui socket, ce rol are fiecare?

Un buffer e de send, altul de receive.

Ce este o întrerupere? Când este livrată o întrerupere?

Basically e un semnal/notificare catre un proces/kernel ca un eveniment exceptional a avut loc. Sunt de doua tipuri: hardware si software. Cele hardware atentioneaza CPU-ul ca un device extern necesita atentie din partea sistemului de operare, iar cele software sunt generate de catre kernel si au drept identificator un nr intreg cu nume simbolice de forma SIGxxxx.

Cu ce diferă port-mapped I/O de memory-mapped I/O?

Cele doua metode de mapare sunt complementare in cadrul I/O. MMIO e mapat in acelasi spatiu de adrese ca programul in sine si e accesat in acelasi mod, iar PMIO are un spatiu de adrese separat si pentru acces se foloseste un set dedicat de instructiuni.

Ce este o operație asincronă?

Operatiile asincrone au drept caracteristica faptul ca pot fi executate in alta ordine fata de cea in care apar in cod.

Ce este o operație neblocantă?

O operatie neblocanta permite executarea altor operatii, chiar daca inca nu a ajuns la finalul executiei.

Cu ce diferă un socket de rețea de un socket UNIX?

Socketii UNIX sunt mecanisme de IPC(interprocess communication) si permit schimb de date bidirectional intre procese care ruleaza in acelasi sistem, in timp de socketii de retea sunt utili in cazul comunicatiei intre procese situate in retele/sisteme diferite.

Care este diferența între un pipe anonim și un pipe cu nume (named pipe)?

Pe langa cea evidenta, adica pipe-ul anonim nu are un nume si este accesibil prin intermediul a doi file descriptori generati de functia pipe(), in timp ce un pipe cu nume primeste numele de la user prin care este referentiat de catre reader/writer, un named pipe poate avea mai multe procese care pot comunica prin acesta, dar unul anonim permite numai comunicarea unidirectionala intre un proces parinte si unul copil.(master-slave)

Ce este buffer cache-ul? Care este rolul său?

La lucrul cu fisiere pe disc, apelurile read/write nu acceseaza direct spatiul de pe disc, ci copiaza date dintr-un buffer din user-space intr-unul din kernel-space denumit ‘buffer cache’. Scopul este de a nu genera timpi morti cauzati de operatii repetate si incete pe disc.

De ce operația write pe fișiere este foarte rar blocantă?

E rar blocanta fiindca procesul de copiere a continutului buffer-ului din user-space in kernel-space dureaza foarte putin. Cazuri in care devine blocanta sunt rare, precum atunci cand nu exista pagini libere si unele trebuie sa fie eliberate si cazul in care write-ul are ca target mijlocul unui fisier, ceea ce presupune citirea continutului din jurul pozitiei.

În ce situație operația read() pe fișier se blochează?

By default read() se blocheaza pana cand cel putin un byte e disponibil spre a fi returnat catre aplicatie, dar e posibil sa ii schimbi comportamentul in non-blocking daca esti suficient de ambitios si ii setezi flag-ul corespunzator file descriptorului (O\_NONBLOCK).

Care este rolul unui device driver?

Este raspunderea device driver-ului sa faca legatura intre buffer-ul din kernel cu controller ul.

Ce rol are controller-ul hardware?

Acest device are un controller in care sunt aceste registre de I/O.

Administreaza corespunzator un periferic (cunoaste specificitatile perifericului respectiv).

Care este rolul DMA-ului (Direct Memory Access)?

DMA-ul faciliteaza transferul de date I/O direct din si in memorie fara interventia procesorului, acesta avand rolul doar de a initia transferul.

Când are sens să folosim polling în loc de întreruperi?

Ar avea sens sa preferam polling-ul daca evenimentele pe care le gestionam sunt sincrone, frecvente(majoritatea ciclurilor de polling genereaza hit-uri) si timpul cat faci polling este limitat(cat mai mic posibil fiindca altfel irosesti cicli de procesare). Altfel, daca evenimentele sunt asincrone, rare si au prioritate crescuta in cadrul aplicatiei preferam intreruperile.

Ce înseamnă zero-copy? Ce mecanism/apel folosește zero-copy?

Zero-copy presupune transferul unei informatii intre 2 buffere fara trecerea prin user-space.

Linux: splice(), sendfile()

Windows: TransmitFile()

Ce rol are mecanismul de TCP offload engine?

Rolul sau este de a usura munca CPU-ului in a gestiona protocolul TCP/IP. TOE-urile sunt concepute de producatorii de placi de retea, fiind implementate direct in hardware. In Linux nu prea ai suport pentru asa ceva deoarece conceptul paraseste ideea de a conferi kernel-ului accesul permenent si total la intreg setul de resurse al sistemului at any time.

Care este sursa primară pentru care un apel send() pe un socket TCP se blochează?

TCP - send se blocheaza ptc receiverul are bufferul plin

Care este sursa primară pentru care un apel send() pe un socket UDP se blochează?

Se umple bufferul de send daca trimite prea mult prea repede.

Ce garanții ni se oferă în momentul în care apelul send() se întoarce în user space?

Datele au fost copiate intr-un send buffer.

Cu ce diferă afișarea folosind printf() față de folosirea write()?

Cred că se referă la buffering - write scrie direct ce i-ai dat, printf face buffering

https://stackoverflow.com/questions/21084218/difference-between-write-and-printf

De ce subsistemul de networking nu folosește buffer cache-ul?

Ce rol are apelul / comanda sync?

The sync command forces an immediate write of all cached data to disk.

Sync muta din buffer cache-ul din kernel space pe disk. Noi cand facem write facem write in

acest buffer cache, nu pe disk direct. Apelul sync duce datele din acest buffer pe disk

Care este rolul apelului ioctl / DeviceIoControl?

IOCTL . IOCTL is referred as Input and Output Control, which is used to talking to

device drivers. This system call, available in most driver categories. The major use of

this is in case of handling some specific operations of a device for which the kernel does

not have a system call by default.

De ce în general doar utilizatorul root are permisiuni de scriere (uneori doar root are permisiuni de citire) pe intrările din /dev?

Pentru ca accesul direct la hardware se face doar din modul privilegiat

e tip bloc de un dispozitiv de tip caracter? Dați câte un exemplu de fiecare.

De ce nu are sens operația de seek pe un dispozitiv de tip caracter?

Operatia de seek n-are sens pentru un dispozitiv de tip caracter deoarece tipul asta de device are o singura pozitie, cea curenta(fiindca datele vin sub forma de stream ‘one character at a time’), asadar nu putem sa miscam un pointer in fata sau inapoi in fisier.

‘sudo cat /dev/input/mice’ you’ll see why

Ce adresă IP locală și ce port local are un socket întors de apelul accept()?

int accept(int sockfd, struct sockaddr \*addr, socklen\_t \*addrlen);

Functia returneaza file descriptor-ul socket-ului nou creat, care contine adresa/port-ul asociat peer-ului (aka aplicatia careia ii permitem sa se conecteze)

Ce valoare poate întoarce un apel read() sau un apel write()?

Un apel de sistem read() sau write() intoarce numarul de bytes cititi/scrisi sau 0 in cazul

end-of-file la read().

Ce operații se pot face pe fișiere?

Operatiile care sunt definite in structura de file\_operations din cadrul kernelului, din care se pot aminti cateva mai populare: read(), llseek(), poll(), ioctl(), mmap(), write() etc.

Avem user space buffer si kernel buffer cu rolul de a limita numarul de accesari directe pe fisierele de pe disc.

Ce operații asupra fișierelor modifică/nu modifică valoarea cursorului unui fișier?

Nu modifica cursorul: operatiile de schimbare ale atributelor unui fisier, operatiile de open/close/create.

Modifica cursorul: operatiile de pozitionare intr-un fisier(obviously), scrierea/citirea si trunchierea.

Ce operații asupra fișierelor modifică/nu modifică dimensiunea fișierului?

Modifica: write daca scrii peste dimensiunea curenta, truncate

Nu modifica: lseek, read

Unde este reținută valoarea cursorului de fișiere (file pointer) și unde este reținută dimensiunea fișierului?

Dimensiunea fisierului e retinuta in structura inod-ului(pe langa owner id, group id, flag-uri de sistem/user, device id etc), iar file pointer-ul rezida in cadrul kernel-ului, acesta avand rolul de I/O managing in File Descriptor Table. (le gasesti de obicei la /proc/\_\_PID\_\_/fd)

De ce avem două buffere asociate fiecărui socket, ce rol are fiecare?

Un buffer e de send, altul de receive.

Ce este o întrerupere? Când este livrată o întrerupere?

Basically e un semnal/notificare catre un proces/kernel ca un eveniment exceptional a avut loc. Sunt de doua tipuri: hardware si software. Cele hardware atentioneaza CPU-ul ca un device extern necesita atentie din partea sistemului de operare, iar cele software sunt generate de catre kernel si au drept identificator un nr intreg cu nume simbolice de forma SIGxxxx.

Cu ce diferă port-mapped I/O de memory-mapped I/O?

Cele doua metode de mapare sunt complementare in cadrul I/O. MMIO e mapat in acelasi spatiu de adrese ca programul in sine si e accesat in acelasi mod, iar PMIO are un spatiu de adrese separat si pentru acces se foloseste un set dedicat de instructiuni.

Ce este o operație asincronă?

Operatiile asincrone au drept caracteristica faptul ca pot fi executate in alta ordine fata de cea in care apar in cod.

Ce este o operație neblocantă?

O operatie neblocanta permite executarea altor operatii, chiar daca inca nu a ajuns la finalul executiei.

Cu ce diferă un socket de rețea de un socket UNIX?

Socketii UNIX sunt mecanisme de IPC(interprocess communication) si permit schimb de date bidirectional intre procese care ruleaza in acelasi sistem, in timp de socketii de retea sunt utili in cazul comunicatiei intre procese situate in retele/sisteme diferite.

Care este diferența între un pipe anonim și un pipe cu nume (named pipe)?

Pe langa cea evidenta, adica pipe-ul anonim nu are un nume si este accesibil prin intermediul a doi file descriptori generati de functia pipe(), in timp ce un pipe cu nume primeste numele de la user prin care este referentiat de catre reader/writer, un named pipe poate avea mai multe procese care pot comunica prin acesta, dar unul anonim permite numai comunicarea unidirectionala intre un proces parinte si unul copil.(master-slave)

Ce este buffer cache-ul? Care este rolul său?

La lucrul cu fisiere pe disc, apelurile read/write nu acceseaza direct spatiul de pe disc, ci copiaza date dintr-un buffer din user-space intr-unul din kernel-space denumit ‘buffer cache’. Scopul este de a nu genera timpi morti cauzati de operatii repetate si incete pe disc.

De ce operația write pe fișiere este foarte rar blocantă?

E rar blocanta fiindca procesul de copiere a continutului buffer-ului din user-space in kernel-space dureaza foarte putin. Cazuri in care devine blocanta sunt rare, precum atunci cand nu exista pagini libere si unele trebuie sa fie eliberate si cazul in care write-ul are ca target mijlocul unui fisier, ceea ce presupune citirea continutului din jurul pozitiei.

În ce situație operația read() pe fișier se blochează?

By default read() se blocheaza pana cand cel putin un byte e disponibil spre a fi returnat catre aplicatie, dar e posibil sa ii schimbi comportamentul in non-blocking daca esti suficient de ambitios si ii setezi flag-ul corespunzator file descriptorului (O\_NONBLOCK).

Care este rolul unui device driver?

**Este raspunderea device driver-ului sa faca legatura intre buffer-ul din kernel cu controller ul.**

Ce rol are controller-ul hardware?

**Acest device are un controller in care sunt aceste registre de I/O.**

Administreaza corespunzator un periferic (cunoaste specificitatile perifericului respectiv).

Care este rolul DMA-ului (Direct Memory Access)?

DMA-ul faciliteaza transferul de date I/O direct din si in memorie fara interventia procesorului, acesta avand rolul doar de a initia transferul.

Când are sens să folosim polling în loc de întreruperi?

Ar avea sens sa preferam polling-ul daca evenimentele pe care le gestionam sunt sincrone, frecvente(majoritatea ciclurilor de polling genereaza hit-uri) si timpul cat faci polling este limitat(cat mai mic posibil fiindca altfel irosesti cicli de procesare). Altfel, daca evenimentele sunt asincrone, rare si au prioritate crescuta in cadrul aplicatiei preferam intreruperile.

Ce înseamnă zero-copy? Ce mecanism/apel folosește zero-copy?

Zero-copy presupune transferul unei informatii intre 2 buffere fara trecerea prin user-space.

Linux: splice(), sendfile()

Windows: TransmitFile()

Ce rol are mecanismul de TCP offload engine?

Rolul sau este de a usura munca CPU-ului in a gestiona protocolul TCP/IP. TOE-urile sunt concepute de producatorii de placi de retea, fiind implementate direct in hardware. In Linux nu prea ai suport pentru asa ceva deoarece conceptul paraseste ideea de a conferi kernel-ului accesul permenent si total la intreg setul de resurse al sistemului at any time.

Care este sursa primară pentru care un apel send() pe un socket TCP se blochează?

TCP - send se blocheaza ptc receiverul are bufferul plin

Care este sursa primară pentru care un apel send() pe un socket UDP se blochează?

Se umple bufferul de send daca trimite prea mult prea repede.

Ce garanții ni se oferă în momentul în care apelul send() se întoarce în user space?

Datele au fost copiate intr-un send buffer.

Cu ce diferă afișarea folosind printf() față de folosirea write()?

Cred că se referă la buffering - write scrie direct ce i-ai dat, printf face buffering

https://stackoverflow.com/questions/21084218/difference-between-write-and-printf

De ce subsistemul de networking nu folosește buffer cache-ul?

Ce rol are apelul / comanda sync?

The sync command forces an immediate write of all cached data to disk.

Sync muta din buffer cache-ul din kernel space pe disk. Noi cand facem write facem write in

acest buffer cache, nu pe disk direct. Apelul sync duce datele din acest buffer pe disk

Care este rolul apelului ioctl / DeviceIoControl?

IOCTL . IOCTL is referred as Input and Output Control, which is used to talking to

device drivers. This system call, available in most driver categories. The major use of

this is in case of handling some specific operations of a device for which the kernel does

not have a system call by default.

De ce în general doar utilizatorul root are permisiuni de scriere (uneori doar root are permisiuni de citire) pe intrările din /dev?

Pentru ca accesul direct la hardware se face doar din modul privilegiat