FISIERE

1)Cum identifica calculatorul tipurile de fisiere?

a)extensia fisierului(.txt, .docx etc.)

b)magic numbers=primii cativa biti ai unui fisier contin aproape intotdeauna o semnatura unica(png are 89 50 4E 47, pdf are %pdf, jpeg are ff d8 ff)

2)Tipuri de fisiere=”totul este un fisier” adica sistemul de operare trateaza totul de la documente si directoare, pana la hard diskuri, tastaturi si conexiuni de retea ca un fisier

a)fisiere obisnuite(container care contine date; aceste date pot fi text, date binare precum imagini, mp3, un program executabil)

b)directoare=tip special de fisier cu functia sa detina o lista de nume ale altor fisiere(e ca un dosar, organizeaza si tine la un loc alte fisiere si dosare). Doar kernelul poate scrie in ele, de exemplu cand creezi un fisier txt folosind touch, procesul touch nu scrie direct in director, ci face o cerere catre kernel pentru a se asigura ca ai permisiunea de a scrie in acel director(pentru a previne coruperea structurii sistemului de fisiere)

c)fisiere de tip device=nu contin date stocate, ci sunt ca niste punti de legatura directa catre o componenta hardware. Programele pot scrie si citi din aceste fisiere pentru a controla HW, iar r/w se face prin intermediul driverelor. Pot fi de tip block device(hard disk de ex) sau character device(tastatura)

d) fifo= spre deosebire de pipe-uri care leaga iesirea standard a unui proces direct la intrarea standard a unui proces, lucru care functioneaza doar cu procese INRUDITE(gen proces parinte->proces copil), un fifo rezolva aceasta limitare, doarece oricare doua procese fara nici o legatura pot comunica cu el, iar datele sunt procesate first in first out

e)socket=un mecanism IPC pentru comunicarea bidirectionala

-canal de comunicatie local(functioneaza ca un fifo dar mai coplex, de exemplu un server de baze de date care ruleaza local si o aplicatie web)

-socket tcp/ip = adica socket de retea care este legat de o adresa ip si un port->permite comunicarea bidirectionala intre masini aflate oriunde in lume

f) link simbolic= scurtatura catre un fisier (folosesc ln -s cale_lunga link_simbolic)

3)Permisiuni de acces= nu vrei sa lasi pe oricine sa iti stearga fisierele sau sa iti citeasca documentele private

Sunt trei niveluri de acces care raspund la intrebarea CINE?:

a)user=esti proprietarul fisierelui, ai cel mai mare control

b)group=este un membru al grupului caruia ii apartine fisierul

c)others= nu esti nici proprietar, nici parte dintr-un grup ~un strain, mai exact

Deci sistemul de operare intreaba: esti proprietarul?(adica uid-ul tau se potriveste cu uid-ul fisierului?) daca da, atunci se aplica permisiunile de utilizator iar restul sunt ignorate;

esti parte din grup?(gid-ul tau se potriveste cu gid-ul fisierului) —//— aceeasi idee

altfel, ti se aplica permisiunile de la altii

Acum, CINE are permisiuni si CARE sunt acestea: read(poti deschide si citi continutul fisierului), write(poti modifica sau sterge continutul fisierului), execute(poti rula fisierul ca pe un program)

Observatii: ai nevoie de w si x pe un director pentru a putea crea fisiere in el, iar pentru a sterge un fisier nu ai nevoie de W pe fisierul respectiv ci ai nevoie de W pe directorul care il contine(plus read daca nu stii EXACT numele fisierului)

4) set UID si set GID= exceptii de la regulile permisiunilor de baza, raspunde la intrebarea ce se intampla cand un utilizator normal trebuie sa execute o actiune care necesita privilegii superioare (de root sau de un alt grup), dar doar pentru o sarcina f specifica precum sa-si schimbe propria parola

asa ar trbuie sa arate permisiunile unui fisier care are setat bitul de set: rws—---in loc de rwx

Fluxul Execuției:
Utilizatorul andrei (UID 1001) tastează comanda passwd.

Kernelul (sistemul de operare) încarcă programul /usr/bin/passwd.

Kernelul vede că acest fișier este deținut de root (UID 0) și are bitul Set-UID (s) activat.

Kernelul pornește procesul passwd, dar în loc să-i dea identitatea lui andrei (UID 1001), îi dă identitatea efectivă a proprietarului fișierului, adică root (UID 0).

Acum, programul passwd rulează ca și cum ar fi fost pornit de root.

Deoarece rulează ca root, programul are permisiunea de a scrie în fișierul /etc/shadow și de a-ți actualiza parola.

Programul passwd își termină treaba și se închide.

Procesul dispare, iar tu, andrei, te întorci în shell-ul tău, care rulează în continuare cu identitatea ta normală.

5)unmask=raspunde la intrebarea: cand creez un fisier nou, ce permisiuni NU vreau sa aiba in mod implicit? - fara unmask, fiecare fisier ar avea permsiune totale(rwxrwxrwx -777)

deci fluxul ar fi: creez un director nou cu permisiunile 777, iar eu am setat unmask 022=> rezultatul final pentru acel director e 755

PROCESE=un program in executie; mai exact programul in sine este un fisier pasiv aflat pe disc si nu face nimic de unul singur(de exemplu chrome.exe), deci procesul apare atunci cand programul este incarcat in memoria ram si executat de procesor=> in acest moment sistemul de operare ii aloca un spatiu de memorie propriu, un pid si resurse

Un proces executa instructiunile una dupa alta.

Dupa cum am zis fiecarui proces i se aloca un spatiu de memorie propriu unde o sa se afle sectiunile: text(codul programului copiat de pe hard disk unde se afla programul), registrele cpu(starea curenta) mai exact memoria de scurta durata care tine minte exact la ce

instructiune a ramas, stiva(contine datele temporare de exemplu variabilele locale, parametrii unei functii, adrese de retur) care functioneaza pe principiul LIFO adica atunci cand chemi o functie datele ei sunt puse deasupra pe stiva, iar cand functia se termina, datele ei sunt date jos de pe stiva si se intoarce prin adresa de retur, DATE(date globale initializate sau nu), HEAP(contine memoria alocata dinamic in timpul rularii gen cu new, malloc)

6) COMUNICARE INTER PROCESS= am doua procese care lucreaza impreuna, acestea trebuie sa sa comunice si sa-si coordoneze munca, cum are loc acest lucru?
Exemple: un chelner preia comanda si partajeaza aceasta informatie bucatarului. Daca bucatarul ar face ambele lucruri atunci procesul ar fi mult mai lent, de asta e necesar sa ai mai multe procese care comunica intre ele, iar fiecare dintre ele este axata pe un lucru specific.

Aceasta comunicare are nevoie de IPC, iar sistemul de operare are doua metode principale pentru a face asta:

a) memorie partajata= ca o tabla alba pe peretele bucatariei(zona de memorie speciala)=> chelnerul si bucatarul au permisiuni de a scrie si de a citi de pe aceasta tabla=> chelnerul scrie am luat comanda, bucatarul citeste. Problema este la sincronizare de exemplu cum ar fi sa se stearga aceeasi zona de pe tabla in exact acelasi timp (adica segmentation fault si warning cu double delete)?Avantajul este ca e foarte rapid in schimb
b) schimb de mesaje=nu mai exista tabla alba, iar chelnerul trebyie sa vorbeasca mai intai cu managerul(kernelul), iar kernelul ii parseaza informatiile bucatarului. Este mai lent deoarece datele trebuie copiate de la procesul A la SO si apoi de la SO la procesul B, DAR ofera siguranta f mare

Analogie: Am procesul A si procesul B, iar singura modalitate de comunicare intre ele este prin posta(sist de operare).(deci nu au memorie partajata)
operatiile de baza sunt send si receive; send(mesaj) -> in casuta postala
receive(mesaj)-> procesul B verifica cutia postala si primeste scrisoarea

Intrebare: Cum stie postasul unde sa duca scrisoarea?

a) procesele isi cunosc identitatea= comunicare directa(ex. sockets)
b) comunicare indirecta=folosesc o casuta postala(intermediar neutru) iar procesele trebuie doar sa stie adresa acelei casute postale. este foarte flexibil pentru ca procesului B nu ii pasa cine trimite informatia, iar procesul A poate fi inlocuit oricand cu un proces C

Cum se sincronizeaza procesele?

a) modul blocant(sincron) procesul A nu isi continua treaba pana cand procesul B trimite un semnal ca a primit scrisoarea, iar tu ca proces B stai si astepti in fata cutiei postale pana cand primesti o scrisoare, altfelul nu poti pleca sa faci altceva
b) mod neblocant= daca exista scrisoare o iei si pleci, altfel iti continui treaba
c) modul rendezvous= cand send si receive sunt blocante=> mesajul este transferat daca sunt ambii prezenti in acelasi timp, iar dupa transfer isi pot continua treaba