-modul in care SO interactioneaza din perspectiva utilizatorului sau din perspectiva sistemului

1. din perspectiva utilizatorului

-In primul rand se face accesul utilizatorului in sistem(reamintim ca dupa procesul de boot primul proces care porneste si este tatal tuturor celorlalte procese este init care este responsabil de initializarea sistemului la un runlevel specific), dupa care init lanseaza procese getty pentru fiecare terminal(rolul sau este sa monitorizeze si sa afiseze promptul de LOGIN)

-In al doilea rand, are loc procesul de login (dupa ce utilizatorul isi introduce numele/username-ul si isi introduce parola, getty cauta in etc/shadow, etc/passwd pentru a vedea daca aceasta corespune) iar daca acesta este valid, getty realizeaza sarcini critice precum setarea variabilelor de mediu si setarea permisiunilor, iar dupa, procesul login lanseaza un interpretor de comenzi conectat la terminalul alocat anterior(shell-ul devine in acest moment procesul principal al sesiunii utilizator)

-in ultimul rand, shell-ul asteapta comenzi de la utilizator(comenzi incorporate in shell gen cd, pwd etc.; fisiere de scripting, sau programe executabile(fisiere binare)). Aceste comenzi apeleaza serviciile sistemului de operare (gestiune a fisierelor, controlul proceselor, administrarea sistemului gen conf. retea, montarea dispozitivelor)

2. din perspectiva sistemului= adica felul in care SO ofera serviciile cerute de programele lansate de utilizator

cum sunt accesate aceste servicii?

a) in mod direct fac apel la kernel (spatiul privilegiat) prin system calls (reamintesc ca kernelul gestioneaza resursele hardware si implementeaza toate serviciile fundamentale)
b) in mod indirect din userspace (userspace este un spatiu neprivilegiat si restrictionat din sistem, adica aici se afla programele tale obisnuite, servere/demoni, in plus in userspace nu poti accesa direct resursele hw sau structurile critice ale kernelului, ca in cazul in care un program din userspace da eroare sau se blocheaza, sa nu mi se blocheze intregul sistem ci doar procesul respectiv) care in cele din urma apeleaza tot la kernel

Servere/ demoni

Server este notiunea generala a tuturor programelor(din userspace, foarte rar in kernel cel putin nu in modul tipic Unix), care ofera un serviciu

Demon este un proces care ruleaza in fundal si nu este asociat vreunui terminal sau interfete grafice, de obicei, demonii sunt lansati la pornirea sistemului si asteapa sa fie solicitati (ex: demonul httpd care asteapta tot timpul cereri http in portul 80, syslogd care logheaza toate mesajele de sistem)

Toti demonii sunt servere, dar nu toate serverele sunt demoni!
Serverele sunt accesibile utilizatorilor prin IPC!

*/ Ce sunt alea apeluri sistem (syscalls)?

Syscall urile sunt modalitatea prin care un program din userspace poate solicita un serviciu de kernel. Syscall urile NU sunt servere, ci sunt o INTERFATA/mecanism de comunicare, implementate in kernel, cu scopul de a oferi functionalitati de baza precum operatii de I/O, crearea de procese, alocarea de memorie etc. deoarece programele din userspace sunt neprivilegiate deci NU pot face aceste lucruri pe cont propriu. */

Servicii de sistem =programe esentiale care ruleaza in fundal pentru a asigura functionarea corecta si modul de executie al sistemului

Acestea sunt pornite la bootarea sistemului, mai exact init le lanseaza

Sunt 2 categorii de servicii de sistem: clasice gen servere/demoni care asteapta si raspund solicitarilor utilizatorului, sau programe de suport care nu raspund la cereri externe(servicii de logare, gestionare a memorie etc.)

Cum sunt furnizate aceste servicii?
In mod direct prin servere/demoni, sau indirect prin asigurarea mediului de executie(servicii care ruleaza incontinuu si se asigura ca programele au resursele necesare pentru a rula de ex serviciul de retea care asigura ca placa de retea este configurata, permitand apoi programelor sa foloseasca socketuri pentru comunicare).

Serviciile de retea sunt configurate in functie de runlevel.

Cum pot sa accesez si gestionez serviciile de sistem? (s-a dat anul trecut la examen)

sunt 3 interfecte de acces si gestiune a serviciilor de sistem: system v, upstart si systemd

1. interfata systemV

Este un sistem de initializare bazat pe conceptul de runlevel care defineste cum opereaza sistemul

In primul rand, avem fisierul /etc/inittab care este punctul central de configurare= procesul init cu pid=1 citeste acest fisier imediat dupa boot. Init citeste aici runlevel-ul default(initdefault). Acest fisier mai contine de asemenea linii care definesc ce procese trebuie pornite in functie de runlevel-ul curent=> serviciile sunt grupate si organizate in functie de runlevel-ul care trebuie sa ruleze, mai exact sunt directoarele rc (run commands) dedicate pentru fiecare runlevel(/etc/rc1.d pentru runlevel 1 etc.), plus unul pentru etapa initiala (/etc/rcS.d unde sunt serviciile de sistem executate in etapa initiala a boot-ului inainte de a intra in runlevel-ul implicit)<=Toate aceste directoare nu contin programele executabile ale serviciilor, ci numai link-uri simbolice catre scripturile de initializare(care se gasesc in directorul central /etc/init.d)

mai exact init=>inittab(citesc ce trebuie de aici)=>director pt runlevel-ul specific care contine link-uri simbolice catre serviciile runlevelului specific

Acum, legat de aceste servicii din /etc/init.d , acestea sunt programele shell care implementeaza logica de control pentru fiecare serviciu (pornire, oprire si reincarcare)
Logica de executie e asta: cand sistemul intra intr-un runlevel nou, procesul init parcurge directorul corespunzator si executa link-urile simbolice. Scriptul din /etc/init.d este apelat de 2 ori, o data cu argumentul stop pentru a opri serviciile care nu sunt necesare in noul runlevel si o data cu start ca sa porneasca serviciile specifice

Structure link-ului simbolic sysVinit

acesta este mecanismul prin care sistemul sysVinit decide ce serviciu sa porneasca/opreasca si cand sa faca acest lucru
[START/STOP][NN][NUME_SERVICIU]

prima litera(S/K) indica ce trebuie sa faca init asupra scriptului de servicu din /etc/init.d
Daca prima litera e S, init stie ca trebuie sa apeleze scriptul respectiv cu argumentul start: ex link-ul simbolic S20apache2 determina executia comenzii /etc/init.d/apache2 cu argumentul start
la fel cand prima litera e K(kill)

Ordinea: [NN]
Cifrele de la mijloc (de la 01 la 99) sunt folosite pentru a controla ordinea de executie mai exact sistemul init parcurge directorul runlevel-ului specific si executa link-urile simbolice in ordine lexicografica, iar logica acestor numere de ordin este ca serviciile care nu depind de nimic pornesc cu numere mici, de exemplu serviciile care depind de alte servicii pornesc cu numere mari deoarece s90apache2 porneste dupa ce reteaua s05network si serviciul dns sunt deja active.

Scripturi init.d

Aceste scripturi sunt programe shell standard care includ un header de configurare pentru a fi gestionate automat
Headerul de configurare este o sectiune de metadate(date despre date) pentru ca programele de gestiune precum update-rc.d sau chkconfig o citesc pentru a intelege cum ar trebui sa se comporte un serviciu: sunt citite si interpretate de programele de gestiune care automatizeaza crearea link-urilor simbolice (alea cu S si K din directoarele specifice runlevel-ului)

asta e structura de inceput a fiecarui script:
```
### BEGIN INIT INFO
# {Keyword}: arg1 [arg2 …]
### END INIT INFO
```
Cum are loc aceasta gestiune automata? ei bine prin keywords: de exemplu $ default-start: [runlevels] specifica in ce runlevels serviciul trebuie sa porneasca automat, mai exact atunci cand folosesc o comanda de automatizare precum update-rc.d acesta va citi aceasta linie si va crea un link simbolic de tip S in directoarele rc[runlevel].d

2. interfata Upstart= trece de la modul ce functionare clasic a lui systemV ce presupune runlevels, la un model care are la baza evenimente(event-driven)

Daca la system V aveam filosofia ca porneste toate serviciile la boot si lasa-le sa ruleze, la Upstart se bazeaza pe filosofia “porneste un serviciu doar ca raspuns la o schimbare de stare/eveniment” gen serviciul nu porneste doar pentru ca este intr-un runlevel specific, ci porneste de exemplu daca ai nevoie de el. Acest model permite sistemului să se inițializeze mult mai rapid, deoarece procesele nu așteaptă unul pe altul (rulare paralelă), și permite serviciilor să fie lansate **doar când survine condiția necesară**, economisind resurse.

Un eveniment este un **mesaj** trimis (sau "emis") în sistem, semnalând că o anumită condiție a fost îndeplinită (ex.: placa de rețea este funcțională, sistemul de fișiere este montat).

continuare tutoriatul urmator…