Upstart, spre deosebire de system v, este bazat pe evenimente, mai exact, fiecare serviciu(job) este intr-o stare de asteptare si reactioneaza la schimbari de stare ale sistemului(mesaje). Aceste mesaje sunt sub forma de stringuri care sunt trimise procesului init, si declanseaza actiuni.

-Principiul de bază: Spre deosebire de execuția secvențială a scripturilor din System V, Upstart nu rulează o listă fixă. El stă într-o stare de așteptare și reacționează la schimbări de stare ale sistemului.
-Evenimentele ca mesaje: Un eveniment în Upstart este un mesaj de notificare trimis către procesul init (PID 1). Aceste mesaje semnalează o schimbare (ex: montarea unui sistem de fișiere, conectarea unui dispozitiv USB, pornirea unui alt serviciu).
-Natura evenimentelor: Evenimentele nu conțin cod executabil; sunt doar semnale (string-uri) care declanșează acțiuni.

Job-urile

Deci serviciile sunt denumite ca fiind joburi in Upstart, si acestea se regasesc in /etc/init cu extensia .conf; Un alt lucru important e sa retinem ca aceste fisiere conf au precedenta mai mare decat orice script utilizat de system V, adica daca sistemul nostru regaseste fisiere de conf si scripturi system V, atunci acesta va ignora complet scripturile. Kernelul utilizeaza subsistemul iNotify pentru a monitoriza directorul init in timp real, astfel, orice modificare/creare/stergere a vreunui fisier conf, acestea sunt detectate imediat, iar configuratia interna a Upstart este actualizata imediat fara a fi necesar sa repornesti sistemul.

Structura Fișierelor de Configurare

Un fișier .conf definește ciclul de viață al unui job prin directive specifice:
-Declanșatori (Start on / Stop on): Aceste directive specifică evenimentele care cauzează pornirea sau oprirea job-ului.
    - Exemplu: start on runlevel [2345] instruiește job-ul să pornească atunci când sistemul intră în modurile multi-user.
    - Exemplu: stop on runlevel [!2345] instruiește job-ul să se oprească dacă sistemul părăsește aceste niveluri (ex: la shutdown sau reboot).
-Execuția (exec / script):
    - exec: Specifică binarul exact care trebuie rulat (ex: exec /sbin/getty -8 38400 tty1).
    - script: Permite blocuri de cod shell pentru logica mai complexă.
-Gestiunea procesului (respawn):
    - Directiva respawn instruiește Upstart să repornească automat procesul dacă acesta se termină neașteptat (exit code diferit de 0 sau terminare prin semnal). Aceasta este critică pentru servicii esențiale (ex: terminale tty, servere web).

Ciclul de Funcționare și Propagarea Evenimentelor

Upstart funcționează printr-un lanț de dependențe rezolvate dinamic:

1. Boot (Startup): La inițializare, Upstart emite evenimentul startup.
2. Activare: Job-urile configurate cu start on startup sunt lansate.
3. Emiterea stărilor intermediare: Pe măsură ce un job își schimbă starea, Upstart emite automat noi evenimente standard asociate acelui job:
    - starting: Emis chiar înainte ca procesul job-ului să fie executat.
    - started: Emis după ce procesul a pornit și rulează (PID-ul este activ).
    - stopping: Emis când se primește cererea de oprire.
    - stopped: Emis după ce procesul s-a terminat.
4. Înlănțuirea: Alte job-uri pot fi configurate să asculte aceste evenimente (ex: un server web poate avea start on started networking). Astfel, sistemul pornește serviciile în paralel, imediat ce dependențele lor sunt satisfăcute.

Modul "User Session"

Upstart permite instanțierea sa nu doar ca PID 1 (system init), ci și ca gestionar de sesiune pentru utilizatori individuali.
-Execuție: Se rulează ca un proces obișnuit cu permisiunile utilizatorului (PID > 1).
Scop: Gestionează servicii specifice utilizatorului (ex: dbus, sesiuni grafice) fără privilegii de root.
-Locații Configurare: Caută fișiere .conf în directoare specifice utilizatorului, precum:

- $HOME/.init/
- $XDG_CONFIG_HOME/upstart
- /usr/share/upstart/sessions (pentru setări globale de sesiune).

Utilitarul de control: initctl

initctl este interfața CLI (Command Line Interface) pentru interacțiunea cu daemonul Upstart.

- initctl list: Afișează toate job-urile înregistrate, starea lor curentă și PID-ul (dacă rulează).
- initctl status <job>: Afișează starea detaliată, bazată pe conceptul de Goal (Scop) și State (Stare):
    - Goal: Ce dorește administratorul să facă job-ul (ex: start, stop).
    - State: Ce face procesul fizic în acel moment (ex: waiting, starting, running, stopping).
- initctl start/stop <job>: Forțează schimbarea stării unui job, ignorând evenimentele start on/stop on.
- initctl emit <eveniment>: Permite generarea manuală a unui eveniment personalizat. Orice job configurat să asculte acel eveniment va reacționa imediat.

Systemd reprezintă standardul actual pentru inițializarea sistemelor Linux, fiind o evoluție majoră față de System V și Upstart prin focusul său pe paralelism și dependențe.

1.def

Spre deosebire de vechile sisteme care rulau liniar (unul după altul), Systemd vede procesul de boot ca pe un arbore de dependențe.

Units

Systemd nu gestionează doar servicii, ci 12 tipuri diferite de obiecte, numite generic Units.

- Definiție: O unitate încapsulează un obiect relevant pentru sistem (un serviciu, un dispozitiv, un punct de montare).
- Stare: Unitățile pot fi active, inactive sau în stări intermediare (activating, deactivating).
- Tipuri principale de unități:
    - Service Units (.service): Controlează demonii (serviciile propriu-zise).
    - Target Units (.target): Grupează alte unități. Sunt folosite pentru sincronizare (ex: "Am ajuns în stadiul în care rețeaua e gata"). Nu fac nimic activ, doar organizează.
    - Device Units (.device): Expun echipamentele hardware în Systemd (permite pornirea unui serviciu doar când un device e conectat).
    - Mount Units (.mount): Gestionează punctele de montare din sistemul de fișiere (înlocuind parțial /etc/fstab).
    - Timer Units (.timer): Activează alte unități la intervale de timp

2. Funcționare și Arhitectură

Procesul de Boot

1. Încărcarea Configurației: Systemd citește fișierele de configurare din două locații principale (cu precedență):
    - /etc/systemd/system/ (Configurări administrator - prioritate maximă).
    - /lib/systemd/system/ sau /usr/lib/systemd/system/ (Configurări implicite ale pachetelor software).
2. Determinarea Tintei (Boot Target): Identifică ținta implicită, de obicei default.target (care este un link simbolic către o țintă reală, ex: graphical.target).
3. Calcularea Arborelui: Determină toate dependențele țintei respective.
4. Activarea: Pornește unitățile. Nu există o ordine strictă (ca la System V), ci o ordine dictată strict de dependențe. Dacă două servicii nu depind unul de altul, vor porni în paralel.

Modul Utilizator (User Session)

- Systemd poate rula instanțe separate pentru fiecare utilizator (PID > 1).
- Comanda: $ systemd --user
- Gestionează servicii specifice sesiunii utilizatorului (ex: server audio, manager de ferestre) fără drepturi de root.

Compatibilitate (Backward Compatibility)

Deși este modern, Systemd păstrează compatibilitatea cu System V:

- Poate rula scripturi vechi din /etc/init.d.
- Oferă interfețe precum /dev/initctl sau /run/initctl (FIFO) pentru a comunica cu uneltele vechi care se așteaptă la Upstart sau SysV.

3. Tinte (Targets) vs. Runlevels

Systemd a înlocuit Runlevel-urile (nivele de rulare numerotate 0-6) cu Targets (Tinte). Tintele sunt mai flexibile deoarece pot fi active simultan.

System V Runlevel	Systemd Target	Descriere
0	poweroff.target	Oprirea sistemului (Shutdown).
1 (Single User)	rescue.target	Mod de recuperare, shell de bază, fără rețea.
3 (Multi-User)	multi-user.target	Sistem complet funcțional, linie de comandă + rețea (fără GUI).
5	graphical.target	multi-user.target + Interfață Grafică (GUI).
6	reboot.target	Repornirea sistemului.

Default Target: Sistemul bootează în ținta către care pointează link-ul simbolic default.target. Aceasta va trage după ea dependențele (ex: networking.service, crond.service).

4. Structura Fișierului de Configurare (.service)

Fișierele Unit sunt inspirate din specificația XDG Desktop Entry (stil .ini). Sunt împărțite în secțiuni:

[Unit]
Description=OpenBSD Secure Shell server After=network.target auditd.service ConditionPathExists=!/etc/ssh/sshd_not_to_be_run
[Service]
EnvironmentFile=-/etc/default/ssh ExecStartPre=/usr/sbin/sshd –t ExecStart=/usr/sbin/sshd -D $SSHD_OPTS ExecReload=/usr/sbin/sshd –t ExecReload=/bin/kill -HUP $MAINPID KillMode=process Restart=on-failure RestartPreventExitStatus=255 Type=notify
[Install]
WantedBy=multi-user.target Alias=sshd.service

A. Secțiunea [Unit]

- Descrie metadatele și relațiile cu alte unități.
- Definește ordinea (ex: After=network.target - pornește doar după ce rețeaua e gata).

B. Secțiunea [Service] (specifică unităților de tip service)

- Conține instrucțiunile efective de execuție.
- ExecStart=...: Comanda pentru pornirea demonului.
- ExecReload=...: Comanda pentru reîncărcarea configurării.
- Aici se definesc politicile de restartare automată (similar cu respawn din Upstart).

C. Secțiunea [Install]

- Nu este folosită la rulare, ci la activare (systemctl enable).
- WantedBy=multi-user.target: Spune sistemului: "Când multi-user.target este activat, activează-mă și pe mine". Creează legătura de dependență inversă.

Reamintim: un demon este un proces care rulează în fundal, fără a fi controlat direct de un utilizator printr-un terminal. **Pentru a transforma un proces in demon** trebuie:

1. Detașarea de terminal

-Scop: Demonul trebuie să fie independent de sesiunea utilizatorului care l-a pornit. deoarece când pornești un program din consolă, el este "fiul" shell-ului (terminalului). Dacă închizi terminalul, shell-ul trimite un semnal (SIGHUP) care omoară toate procesele fiu.
-Acțiune: Prin detașare (uzual folosind fork() și setsid()), procesul devine liderul propriei sesiuni și nu mai depinde de terminalul de control (TTY). Astfel, poate rula indefinit, chiar și după delogarea utilizatorului.

2. Schimbarea directorului de lucru curent

-Acțiune: Demonul execută comanda chdir("/") pentru a se muta în rădăcina sistemului de fișiere (sau într-un alt director dedicat și sigur).
De ce este neaparat necesar?

- Blocarea fișierelor (Umount): Un sistem de fișiere nu poate fi demontat (unmounted) dacă un proces activ are directorul de lucru setat acolo. Dacă demonul ar rămâne în /home/user/, administratorul nu ar putea demonta partiția /home pentru mentenanță fără să omoare demonul.
- Fișiere Core: În cazul unei erori fatale (crash), sistemul de operare poate genera un fișier "core dump" (o imagine a memoriei procesului pentru debugging). Acesta se salvează implicit în directorul de lucru curent. Mutarea în / sau într-un director specific (/var/run/demon) asigură că aceste fișiere nu poluează directoare aleatoare.

3. Închiderea descriptorilor de fișiere deschiși

-În Linux, când un proces părinte (shell-ul) creează un proces fiu (demonul), fiul moștenește toți descriptorii de fișiere deschiși ai părintelui.
-Problemă: Shell-ul poate avea deschise fișiere care nu au nicio legătură cu demonul. A le ține deschise consumă resurse (limita de fișiere deschise) și poate reprezenta un risc de securitate.
-Acțiune: Demonul iterează prin toți descriptorii posibili (de la 3 în sus, deoarece 0, 1, 2 sunt standard) și îi închide preventiv.

4. Redirectarea stdin, stdout și stderr la /dev/null

-Acțiune: Descriptorii standard (0 - intrare, 1 - ieșire, 2 - eroare) sunt conectați la /dev/null (găura neagră a sistemului).
-Mecanism:

- Citire (stdin): Dacă demonul încearcă să citească date, va primi imediat EOF (End Of File). Nu va bloca așteptând input de la o tastatură care nu există.
- Scriere (stdout/stderr): Datele trimise aici sunt aruncate și pierdute, fără a genera erori.

De ce nu îi închidem pur și simplu?

- Multe funcții din bibliotecile standard (ex: printf, perror sau funcții criptografice) presupun că descriptorii 0, 1 și 2 sunt valid deschiși.
- Dacă i-am închide, următorul fișier deschis de demon (ex: un fișier de configurare important) ar putea primi descriptorul 1. Dacă apoi o funcție de bibliotecă face un printf (scriere la descriptorul 1), ar corupe fișierul de configurare scriind text în el. Redirectarea către /dev/null previne acest scenariu.

5. Utilizarea syslogd pentru logarea erorilor

-Problemă: Deoarece am anulat stdout și stderr (la pasul anterior), demonul nu mai poate afișa erori pe ecran. "Ecranele" nu există pentru un proces de fundal.
-Soluție: Demonul folosește facilitatea de sistem Syslog.

Syslogd este serviciul central care colectează mesajele de la diverse aplicații și decide ce să facă cu ele.
-Funcționare: Ascultă mesajele trimise de procesele din sistem.
-Configurare (/etc/syslog.conf sau /etc/rsyslog.conf): Acest fișier dictează regulile de sortare. Pentru fiecare mesaj primit, demonul verifică configurația pentru a decide acțiunea:

- Scriere în fișier: Adaugă mesajul la sfârșitul unui fișier text (ex: /var/log/syslog).
- Afișare pe consolă: Trimite mesajul direct pe /dev/console (util pentru erori critice de kernel).
- Forwarding (Trimitere la distanță): Trimite mesajul prin rețea către un alt server de logare (centralizare).

-Rețea: Serviciul ascultă standard pe portul 514 UDP (definit în /etc/services).
-Administrare:

- La modificarea configurației, nu este necesar restartul complet, ci doar trimiterea semnalului SIGHUP (kill -HUP <PID>) sau folosirea comenzii moderne service rsyslog reload.
- Este pornit și controlat de init sau systemd.

2. Interfața de Logare: syslog() și logger

Aplicațiile nu scriu direct în fișiere (pentru a evita probleme de permisiuni și concurență), ci trimit mesaje către Syslogd prin două metode principale:

1. Funcția syslog(): Folosită în programare (C/C++) de către demoni. Înlocuiește funcțiile standard gen fprintf(stderr, ...).
2. Comanda logger: Folosită în scripturi shell pentru a trimite mesaje către syslogd.
3. Direct pe rețea: Trimiterea de pachete UDP pe portul 514 (mai rar folosit local, uzual pentru echipamente de rețea).

3. Structura Mesajului: Prioritate (Level + Facility)

Pentru ca Syslogd să poată filtra mesajele (să nu amestece erorile critice cu informațiile banale), fiecare mesaj are atașată o "etichetă" numită Prioritate. Aceasta este compusă din două elemente:

A. Nivelul (Level) - Importanța

Indică gravitatea mesajului. Este o valoare numerică între 0 și 7:

- 0 (Emergency): Sistem neutilizabil.
- 1 (Alert): Acțiune necesară imediat.
- ...
- 3 (Error): Erori de funcționare.
- 5 (Notice): Normal, dar semnificativ (Valoarea Implicită).
- 7 (Debug): Informații pentru depanare.

B. Facilitatea (Facility) - Sursa

Indică tipul programului care a generat mesajul. Permite separarea log-urilor în fișiere diferite în funcție de sursă.

- Exemple:
    - kern: Mesaje de la kernel.
    - auth: Autentificare (login, su, sudo).
    - cron: Planificator de sarcini.
    - mail: Serverul de mail.
    - lpr: Sistemul de imprimare.
    - local0 - local7: Facilități rezervate pentru uz personalizat/local.
- Valoarea Implicită: LOG_USER (mesaje generice de utilizator).

Crond este un demon (serviciu de fundal) responsabil cu execuția planificată a comenzilor la momente specifice de timp. Spre deosebire de alte servicii care așteaptă evenimente de rețea sau hardware, Crond așteaptă timpul.

- Pornire: Este lansat automat la bootarea sistemului ca un serviciu de sistem.
- Mod Interactiv: Poate fi pornit manual în mod "foreground" (nedemonizat, rămâne atașat terminalului) folosind comanda:
  $ cron -f

Gestiunea Configurațiilor (Tabelele Crontab)

Crond nu știe implicit ce are de făcut; el citește instrucțiuni din fișiere numite crontabs. Acestea sunt încărcate în memorie la pornire.
Există două tipuri de locații pentru aceste tabele:

A. Tabelele Utilizatorilor

- Locație: /var/spool/cron/crontabs/$USER
- Administrare: Nu se editează direct fișierele text! Se folosește utilitarul dedicat crontab (care asigură verificarea sintaxei și notificarea demonului):
  - crontab -e (editare)
  - crontab -l (listare)

B. Tabelele de Sistem

- Locație: /etc/crontab
- Diferență: Spre deosebire de tabelele de utilizator, aici se specifică explicit sub ce user rulează comanda.
- Directoare standard: Pentru simplificare, distribuțiile Linux includ directoare predefinite rulate de /etc/crontab sau anacron:
  - /etc/cron.hourly (rulează scripturile din el o dată pe oră)
  - /etc/cron.daily (zilnic)
  - /etc/cron.weekly (săptămânal)
  - /etc/cron.monthly (lunar)

3. Ciclul de Funcționare (Algoritmul)

Crond este un demon care "se trezește" periodic.

1. Ciclul de 60 de secunde: Demonul are o rezoluție de un minut. La fiecare minut, el se activează și verifică ceasul sistemului.
2. Verificarea Planificării: Parcurge tabelele încărcate în memorie. Dacă o regulă din tabel se potrivește cu minutul, ora, ziua curentă, demonul execută comanda respectivă.
3. Detectarea Modificărilor:
   - În același ciclu de un minut, Crond verifică metadatele (timpul modificării) pentru:
     - Directorul de spool (/var/spool/cron/crontabs)
     - Fișierul de sistem (/etc/crontab)
   - Reîncărcare: Dacă observă că data modificării s-a schimbat, reciteste și reîncarcă în memorie toate tabelele.

    - Notă: Comanda crontab actualizează automat timpul de modificare al directorului de spool, forțând astfel demonul să vadă noua configurare în maxim 60 de secunde.

4. Gestiunea Ieșirilor (Output)

Deoarece sarcinile cron rulează în fundal, nu au un monitor atașat pentru a afișa rezultatele (stdout) sau erorile (stderr).

- Comportament implicit: Crond capturează orice text afișat de comanda executată.
- Notificare prin Email: Trimite acest text prin email către:
    1. Proprietarul tabelei crontab (utilizatorul care a planificat sarcina).
    2. Sau către adresa specificată în variabila MAILTO= definită la începutul fișierului crontab.
- Evitarea spam-ului: Dacă o comandă nu afișează nimic (este "tăcută" sau output-ul este redirectat la /dev/null), nu se trimite niciun email.