UserFS

Drăgunoi Miruna; Panaet Maria-Alexandra

Universitatea din București
Departamentul de Informatică
Instrumente și tehnici de bază în informatică
2024-2025

Descrierea problemei

Context general: Monitorizarea utilizatorilor activi este esențială pentru gestionarea eficientă a resurselor și securitatea în sistemele de operare. Sistemele multi-utilizator au nevoie de soluții clare pentru a organiza și analiza activitățile utilizatorilor.

Date Particulare: Pentru monitorizarea utilizatorilor se crează o structură de directoare care să reprezinte fiecare utilizator activ. Astfel, se face o generare automată a fișierelor ce conțin informații despre procesele active și istoricul sesiunilor utilizatorilor.

În acest scop, proiectul nostru are rolul de a construi o soluție automatizată pentru monitorizarea și actualizarea periodică a diferitelor informații despre utilizatori. Astfel, vom asigura o administrare eficientă a sistemului și detectarea rapidă a potențialelor probleme de utilizare sau securitate.

Specificația soluției

Ca mod de lucru, vom crea pentru fiecare utilizator activ un director rădăcină pentru a centraliza toate informațiile despre aceștia. Folosim două tipuri de fișiere:

- procs -> fișiere dedicate pentru listarea proceselor curente
- lastlogin -> fișiere ce stochează istoricul sesiunilor unui utilizator ce nu mai este activ.

Pentru eficacitate, tot procesul va avea o actualizare automată la fiecare 30 de secunde.

Drept caracteristici ale prototipului, proiectul nostru urmărește un script shell ușor de utilizat și portabil. Performanța sistemului este conferită de faptul că se minimizează impactul asupra acesteia. Modul nostru de abordare este unul mai restrâns, însă este ușor de extins și pentru o serie de sisteme mult mai complexe.

Drept cerințe tehnice urmărite, sistemul de operare pe care vom lucra este Unix/Linux. Vom folosi și diferite comenzi shell în procesul de monitorizare a utilizatorilor, cum ar fi: ps, who, touch, mkdir. Astfel, este nevoie de acces de administrator pentru modificarea structurilor de directoare folosite și utilizarea unor comenzi de sistem.

Desigur, proiectul nostru urmărește și niște presupuneri și diferite constrângeri. Ca exemplu, utilizatorii monitorizați pot fi vizibili prin comanda who. Am luat în calcul și faptul că intervalul de actualizare să fie suficient de mic pentru a menține informațiile precise, dar fără să afecteze performanța sistemului de operare.

Designul soluției

Proiectul nostru, ca arhitectură software, se bazează pe un script principal în shell pentru a gestiona întregul proces de monitorizare. În același timp, se utilizează comenzi shell pentru colectarea și actualizarea datelor.

Structura rezultată este formată din:

- director rădăcină -> conține câte un subdirector pentru fiecare utilizator activ
- subdirectoarele -> fișierele procs și lastlogin specifice fiecărui utilizator

Mecanisme folosite pe parcursul programului proiectului:

- comanda who -> astfel se obțin utilizatorii activi
- comanda ps -> astfel se listează procesele asociate fiecărui utilizator
- buclă continuă -> astfel se realizează actualizările periodice (pentru acest aspect se poate utiliza și un job cron)

Avantajele buclei continue:

- control mai mare asupra execuției (în cazul unui job cron, controlul asupra execuției repetate este limitat)
- interval de timp precis (un job cron poate fi setat doar la intervale mai mari de 1 minut)
- execuție continuă fără suprapuneri (cu cron, dacă scriptul durează mai mult decât intervalul setat, pot apărea suprapuneri)
- programul este mai ușor de testat si modificat (scriptul în buclă poate fi pornit și testat direct într-o sesiune de terminal, fără a fi nevoie de configurarea unui crontab).

Justificarea modului de abordare și de lucru ales: am încercat să folosim o abordare simplă și eficientă pentru majoritatea scenariilor comune. În același timp, scriptul shell oferă flexibilitate și ușurință în întreținere.

Implementare

Componenta software este bazată pe un script shell principal responsabil de întreaga logică de monitorizare și actualizare. Drept unelte folosite în proiectul nostru, avem comenzile shell necesare și esențiale (ps, who, mkdir, touch, echo), dar și fișiere temporare pentru gestionarea intermediară a datelor.

Desigur, pe parcursul realizării proiectului am întâlnit și provocări tehnice. Gestionarea erorilor a intervenit din cauza permisiunilor insuficiente și a proceselor fantomă, pe care le-am rezolvat ulterior. Un alt detaliu ce a fost necesar să-l urmărim a fost sincronizarea datelor între sesiunile consecutive. Pentru eficacitate, am optimizat scriptul pentru a rula eficient pe sisteme și cu mai mulți utilizatori activi.

În același timp, dependența de comenzi specifice sistemelor Unix sau Linux a reprezentat o limitare. De asemenea, performanța algoritmului nu este garantată în cazul în care numărul de utilizatori activi devine unul prea mare.

Experimente

Proiectul nostru folosește un mediu de testare specific pentru cerință, monitorizarea utilizatorilor activi, astfel:

- hardware -> procesor 2-core, memorie 4 GB
- software -> ubuntu 20.04, Bash 5.0

Pentru a valida cât mai eficient programul proiectul nostru, am încercat să folosim un scenariu de test destul de puternic, după cum urmează:

- 5 utilizatori activi simultan în sistemul de operare
- testarea intrării și ieșirii utilizatorilor din sistem
- creșterea treptată a numărului de utilizatori până la 50 pentru a analiza stabilitatea, dar și eficiența programului

Experimente pe matrice

Pentru a analiza performanța scriptului, s-au utilizat matrice pentru a măsura timpul mediu de actualizare în funcție de numărul de utilizatori activi:

Număr utilizatori	Timp Mediu de Actualizare
5	1 secundă
10	1.5 secunde
20	2.5 secunde
50	3 secunde

Scenariile de test includ adăugarea progresivă a utilizatorilor și rularea simultană a proceselor pe fiecare cont.

Rezultatele scenariilor de test utilizate pentru a verifica programul proiectului nostru de monitorizare a utilizatorilor activi sunt următoarele:

- timpul mediu de actualizare este de aproximativ o secundă pentru 5 utilizatori din sistem
- stabilitate menținută până la 50 de utilizatori activi în sistem, cu un timp mediu de actualizare de aproximativ 3 secunde
- fișierele procs și lastlogin au reflectat corect stările utilizatorilor în toate scenariile testate.

Un caz de test specific implementării este monitorizarea unui utilizator fictiv, denumit testuser, care:

1. Intră în sistem cu un script automatizat: folosim comanda -> sudo useradd -m testuser && sudo su - testuser.
2. Rulează câteva procese pentru simulare, cum ar fi: sleep 300 & echo “Test script running...”.
3. Este deconectat din sistem și verificat pentru actualizarea fișierului *lastlogin* -> ```bash pkill -u testuser

Concluzii

Aspectele importante pe care le-am urmărit pe parcursul proiectului se bazează pe găsirea unei soluții ce oferă un mecanism simplu și eficient pentru monitorizarea utilizatorilor activi, dar și un script shell ce este ușor de personalizat și integrat în alte sisteme.

Desigur, există și îmbunătățiri ce pot fi făcute, cum ar fi optimizarea performanței pentru mediile cu resurse limitate, adăugarea unui mod de raportare grafică pentru a analiza istoricul utilizatorilor, dar și extinderea funcționalității pentru a monitoriza și alte resurse (cum ar fi utilizarea memoriei).

În timpul realizării proiectului, am bifat și o serie de aspecte noi pe care le-am învățat:

- importanța utilizării eficiente a scripturilor shell pentru automatizare
- cum să gestionăm structuri dinamice de date în sisteme multi-utilizator
- rolul testării în asigurarea calității soluțiilor software.