PROIECT TEHNIC - SISTEM DE MONITORIZARE ȘI RESTART AUTOMAT PENTRU PROCESE CRITICE

1. OBIECTIVELE PROIECTULUI

1.1 Considerații Generale

Sistemul de monitorizare și restart automat pentru procese critice reprezintă o soluție software avansată destinată să asigure disponibilitatea continuă a serviciilor esențiale într-un mediu Linux. În contextul actual, unde sistemele informatice joacă un rol critic în operațiunile zilnice, este imperios necesar să se implementeze mecanisme automate de detectare și remediere a problemelor care pot afecta funcționarea normală a serviciilor.

1.2 Scop

Proiectul urmărește dezvoltarea unui sistem robust și scalabil care să monitorizeze în timp real starea proceselor critice, să detecteze automat situațiile de alarmă și să execute acțiuni corective prin restartarea serviciilor afectate. Sistemul trebuie să funcționeze independent, să ofere vizibilitate asupra operațiunilor și să mențină un nivel ridicat de disponibilitate a infrastructurii.

1.3 Obiectivul General

Implementarea unui serviciu de monitorizare automată care să detecteze și să remedieze automat problemele de funcționare a proceselor critice, reducând timpul de indisponibilitate și eliminând necesitatea intervenției manuale în majoritatea cazurilor.

2. DESCRIEREA GENERALĂ

2.1 Arhitectura Sistemului

Sistemul este compus din următoarele componente principale:

- **Serviciul de Monitorizare** (`monitor_service.sh`): Scriptul principal care implementează logica de monitorizare
- **Baza de Date MySQL**: Stochează informațiile despre procese și stările lor
- **Sistemul de Configurare** (`config.ini`): Fișier de configurare pentru personalizarea comportamentului
- **Serviciul Systemd** (`monitor_service.service`): Unitatea de serviciu pentru integrarea cu sistemul
- **Sistemul de Logging**: Implementează rotația automată a logurilor și integrarea cu syslog

2.2 Funcționalități Principale

- Monitorizare continuă a proceselor critice
- Detectare automată a stărilor de alarmă
- Restart automat cu strategii configurabile
- Sistem de circuit breaker pentru prevenirea restarturilor repetate
- Health check configurabil pentru fiecare proces

- Logging avansat cu rotație automată
- Interfață CLI pentru administrare

3. CONSIDERAȚII

3.1 Considerații de Securitate

- Utilizarea fișierelor temporare pentru credențialele MySQL
- Implementarea timeout-urilor pentru conexiunile la baza de date
- Restricționarea permisiunilor utilizatorului de baza de date
- Logarea evenimentelor critice în syslog

3.2 Considerații de Performanță

- Optimizarea conexiunilor MySQL cu timeout-uri și compresie
- Implementarea circuit breaker pattern pentru evitarea restarturilor repetate
- Configurarea intervalelor de verificare personalizabile
- Rotația automată a logurilor pentru menținerea performanței

3.3 Considerații de Scalabilitate

- Configurarea flexibilă per proces
- Suport pentru multiple strategii de restart
- Sistem modular de configurare
- Extensibilitatea pentru noi tipuri de procese

4. CERINȚE PRIVIND SOLUȚIA TEHNICĂ

4.1 Cerințe Funcționale

4.1.1 Monitorizare Continuă

- Sistemul trebuie să verifice starea proceselor la intervale configurabile
- Detectarea automată a stărilor de alarmă (alarma=1, sound=0)
- Suport pentru multiple tipuri de procese și servicii

4.1.2 Restart Automat

- Implementarea strategiilor de restart: service, process, custom, auto
- Suport pentru comenzi pre-restart configurabile
- Health check post-restart pentru verificarea succesului
- Retry logic cu număr configurat de încercări

4.1.3 Circuit Breaker Pattern

- Prevenirea restarturilor repetate pentru procesele problematice
- Configurarea pragului de eșecuri și timpului de reset
- Logging și notificări pentru stările de circuit breaker

4.1.4 Logging și Monitorizare

- Rotația automată a logurilor bazată pe dimensiune
- Integrarea cu syslog pentru evenimentele critice
- Nivele de logging configurabile (DEBUG, INFO, WARNING, ERROR, CRITICAL)

4.2 Cerințe Non-Funcționale

4.2.1 Performanță

- Răspuns rapid la detectarea problemelor
- Utilizarea eficientă a resurselor sistemului
- Optimizarea conexiunilor la baza de date

4.2.2 Fiabilitate

- Funcționarea continuă fără intervenție manuală
- Gestionarea corectă a erorilor și excepțiilor
- Recuperarea automată după eșecuri

4.2.3 Mentenabilitate

- Configurarea flexibilă prin fișiere INI
- Logging detaliat pentru debugging
- Documentația completă și exemple de utilizare

4.2.4 Securitate

- Gestionarea sigură a credențialelor
- Validarea configurației la pornire
- Restricționarea permisiunilor de acces

4.3 Cerințe de Configurare

4.3.1 Baza de Date

- Suport pentru MySQL/MariaDB
- Tabele pentru stocarea stării proceselor
- Indexare pentru performanță optimă

4.3.2 Procese și Servicii

- Configurarea individuală per proces
- Suport pentru multiple strategii de restart
- Health check-uri configurabile
- Timeout-uri și delay-uri personalizabile

4.3.3 Logging

- Dimensiunea maximă a fișierelor de log
- Numărul de fișiere de backup
- Rotația automată bazată pe dimensiune

5. IMPLEMENTAREA TEHNICĂ

5.1 Structura Codului

5.1.1 Funcții Principale

- `main()`: Bucla principală de monitorizare
- `get_alarm_processes()`: Interogarea bazei de date pentru procesele în alarmă
- `restart_process()`: Implementarea strategiilor de restart
- `perform_health_check()`: Verificarea stării proceselor după restart
- `check_circuit_breaker()`: Implementarea pattern-ului circuit breaker

5.1.2 Gestionarea Configurației

- `read_config()`: Parsarea fișierului de configurare
- `get_process_config()`: Obținerea configurației specifice proceselor
- Validarea parametrilor de configurare

5.1.3 Sistemul de Logging

- `log()`: Funcția principală de logging
- `check_log_rotation()`: Verificarea necesității rotației
- `rotate_logs()`: Implementarea rotației logurilor

5.2 Strategiile de Restart

5.2.1 Service Strategy

- Utilizarea `systemctl` pentru gestionarea serviciilor
- Verificarea stării serviciului înainte de restart
- Health check prin `systemctl is-active`

5.2.2 Process Strategy

- Utilizarea `pgrep` pentru verificarea proceselor
- Gestionarea directă a proceselor cu `pkill`
- Health check prin `pgrep`

5.2.3 Custom Strategy

- Comenzi personalizate de restart
- Suport pentru substituția parametrilor
- Health check-uri configurabile

5.3 Circuit Breaker Implementation

5.3.1 Stările Circuit Breaker

- **Closed**: Funcționare normală
- **Open**: Restarturile sunt blocate
- **Reset**: Recuperarea după perioada de cooldown

5.3.2 Logica de Decizie

- Numărarea eșecurilor consecutive
- Compararea cu pragul configurat
- Resetarea automată după timpul de cooldown

6. TESTAREA ȘI VALIDAREA

6.1 Scriptul de Testare

- `test_alarm.sh`: Simularea stărilor de alarmă
- Testarea tuturor strategiilor de restart
- Validarea funcționării circuit breaker

6.2 Scenarii de Testare

- Procese care intră în stare de alarmă
- Eșecuri consecutive de restart
- Funcționarea circuit breaker-ului
- Rotația logurilor

7. DEPLOYMENT ȘI OPERAȚIONALIZARE

7.1 Instalarea Sistemului

- Copierea fișierelor în `/opt/monitor_service`
- Configurarea bazei de date cu `setup.sql`
- Instalarea serviciului systemd

7.2 Configurarea Inițială

- Editarea `config.ini` pentru mediul specific
- Configurarea credențialelor bazei de date
- Definirea proceselor de monitorizat

7.3 Operaționalizarea

- Pornirea serviciului cu `systemctl`
- Monitorizarea logurilor
- Configurarea pornirii automate

8. MONITORIZAREA ȘI MENTENANȚA

8.1 Logurile Sistemului

- Monitorizarea `/var/log/monitor_service.log`
- Integrarea cu syslog pentru evenimentele critice
- Rotația automată pentru menținerea performanței

8.2 Comenzile de Administrare

- `--status`: Verificarea stării proceselor
- `--restart`: Restartarea manuală a proceselor
- `--help`: Ajutorul pentru utilizare

8.3 Mentenanța Configurației

- Actualizarea configurației proceselor
- Adăugarea de noi procese de monitorizat
- Modificarea strategiilor de restart

9. AVANTAJELE SOLUȚIEI

9.1 Automatizarea Completă

- Eliminarea intervenției manuale în majoritatea cazurilor
- Răspuns rapid la probleme
- Disponibilitate îmbunătățită a serviciilor

9.2 Flexibilitatea Configurării

- Adaptarea la diferite tipuri de procese
- Strategii de restart personalizabile
- Health check-uri configurabile

9.3 Robustețea Sistemului

- Implementarea circuit breaker pattern
- Gestionarea corectă a erorilor
- Recuperarea automată după eșecuri

9.4 Scalabilitatea

- Suport pentru multiple procese
- Configurarea individuală per proces
- Extensibilitatea pentru noi cerințe

10. LIMITĂRI ȘI CONSIDERAȚII

10.1 Limitări Tehnice

- Dependența de MySQL/MariaDB
- Necesitatea permisiunilor de root pentru restart
- Limitările strategiilor de restart disponibile

10.2 Considerații de Securitate

- Gestionarea credențialelor în fișiere de configurare
- Necesitatea permisiunilor elevate pentru restart
- Logarea informațiilor sensibile

10.3 Considerații de Performanță

- Impactul verificărilor la baza de date
- Utilizarea resurselor sistemului
- Scalabilitatea cu numărul de procese

11. EVOLUȚIA VIITOARE

11.1 Îmbunătățiri Planificate

- Suport pentru multiple baze de date
- Interfață web pentru administrare
- Notificări prin email/SMS
- Integrarea cu sisteme de monitorizare externe

11.2 Extensii Posibile

- Suport pentru containere Docker
- Monitorizarea resurselor sistemului
- Integrarea cu sisteme de orchestrare
- Suport pentru multiple noduri

12. CONCLUZII

Sistemul de monitorizare și restart automat pentru procese critice reprezintă o soluție tehnică robustă și scalabilă care adresează eficient necesitatea de menținere a disponibilității serviciilor critice într-un mediu Linux.

12.1 Realizările Proiectului

- Implementarea unui sistem complet de monitorizare automată
- Dezvoltarea strategiilor flexibile de restart
- Integrarea pattern-ului circuit breaker pentru robustețe
- Crearea unui sistem de logging avansat cu rotație automată
- Implementarea unui sistem de configurare flexibil și extensibil

12.2 Beneficiile Implementării

- Reducerea semnificativă a timpului de indisponibilitate
- Eliminarea necesității intervenției manuale în majoritatea cazurilor
- Îmbunătățirea generală a fiabilității infrastructurii
- Reducerea costurilor de operaționalizare
- Creșterea nivelului de încredere în sistemele informatice

12.3 Valoarea Adăugată

Soluția implementată oferă o valoare semnificativă prin:

- Automatizarea proceselor de remediere
- Implementarea best practice-urilor din domeniul monitorizării
- Crearea unei baze solide pentru extinderea funcționalităților
- Oferirea unei alternative cost-effective la soluțiile comerciale

Sistemul demonstrează că prin implementarea unor principii de design solide, utilizarea pattern-urilor de reziliență și o configurare flexibilă, este posibil să se realizeze un sistem de monitorizare robust care să răspundă eficient cerințelor actuale și să fie pregătit pentru viitoarele provocări tehnice.
