Project SO

Subiectul proiectului este monitorizarea modificarilor aparute in directoare de-a lungul timpului, prin realizarea de capturi (snapshots) la cererea utilizatorului.

Cerintele programului:

- ·Se va dezvolta un program în limbajul C, prin care utilizatorul va putea specifica directorul de monitorizat ca argument în linia de comandă, iar programul va urmări schimbările care apar în acesta și în subdirectoarele sale.
- La fiecare rulare a programului, se va actualiza captura (snapshot-ul) directorului, stocând metadatele fiecărei intrări.

Pentru a reprezenta grafic un director și schimbările survenite înainte și după rularea programului de monitorizare a modificărilor, putem utiliza un arbore de fișiere simplu.

Directorul înainte de rularea programului:

```
Director Principal
|_ Subdirector1
|  |_ Fișier1.txt
|  |_ Fișier2.txt
|_ Subdirector2
|  |_ Fișier3.txt
|_ Fișier4.txt
```

Directorul după rularea programului:

```
// Versiunea 1
Director Principal
|_ Subdirector1
|  |_ Fișier1.txt
|  |_ Fișier2.txt
|_ Subdirector2
|  |_ Fișier3.txt
|_ Fișier4.txt
|_ Snapshot.txt(sau alt format specificat)
```

```
// Versiunea 2
Director Principal
|_ Subdirector1
|  |_ Fișier1.txt
|  |_ Fișier1_snapshot.txt
|  |_ Fișier2.txt
|  |_ Fișier2_snapshot.txt
|_ Subdirector2
|  |_ Fișier3.txt
|  |_ Fișier3_snapshot.txt
|_ Fișier4.txt
|_ Fișier4_snapshot.txt
```

Nota: S-au prezentat două variante ca exemplu, însă organizarea directorului și a fișierelor de snapshot poate fi realizată în orice fel dorit, fără a exista restricții impuse.

Explicatie versiunea 1: După rularea programului, în directorul principal este creat un nou fișier, denumit "Snapshot.txt" (sau alt format specificat), care conține metadatele fiecărei intrări (fișiere și subdirectoare) din directorul principal și subdirectoarele sale. Acest snapshot este actualizat la fiecare rulare a programului pentru a reflecta modificările survenite.

Nota: Selectarea metadatelor este complet la latitudinea utilizatorului. Este important să includem informații care să diferențieze fiecare intrare din director într-un mod unic și să evidențieze modificările efectuate.

S7

- Funcționalitatea programului va fi actualizată pentru a permite primirea unui număr nespecificat de argumente (directoare) în linia de comandă (dar nu mai mult de 10), iar fiecare argument este diferit. Logica pentru capturarea metadatelor va fi acum aplicată tuturor argumentelor primite, ceea ce înseamnă că programul va actualiza snapshot-urile pentru toate directoarele specificate de utilizator.
  De exemplu:

```
./program_exe dir1 dir2 dir3 dir4
```

- Pentru fiecare intrare din directoarele furnizat ca argumente, utilizatorul va putea compara snapshot-ul anterior al directorului specificat cu cel actual. Dacă există diferențe între cele două snapshot-uri, captura veche va fi actualizată cu noile informații din snapshot-ul curent.

```
// Fișierul instantaneu vechi pentru File1.txt
Marca temporală: 2023-03-15 10:30:00
Intrare: File1.txt
Dimensiune: 1024 octeți
Ultima modificare: 2023-03-14 08:45:00
Permisiuni: rw-r--r--
Număr Inode: 12345
```

```
// Fișierul instantaneu nou pentru File1.txt
Marca temporală: 2023-03-16 11:00:00
Intrare: File1.txt
Dimensiune: 1024 octeți
Ultima modificare: 2023-03-16 09:15:00
Permisiuni: rw-r--r-x
Număr Inode: 12345
```

În exemplele de mai sus, avem o reprezentare posibilă a unui snapshot vechi și curent (nou) pentru File1.txt. Având în vedere că noul snapshot este diferit de cel anterior, cel vechi va fi suprascris.

- Funcționalitatea codului va fi extinsă astfel încât programul să primească un argument suplimentar, reprezentând directorul de ieșire în care vor fi stocate toate snapshot-urile intrărilor din directoarele specificate în linia de comandă. Acest director de ieșire va fi specificat folosind opțiunea `-o`.
  De exemplu, comanda pentru a rula programul va fi:

```
./program_exe -o director_iesire dir1 dir2 dir3 dir4
```

Proiect SO Saptamana 8

Dezvoltați logica de creare a snapshot-ului pentru fiecare director analizat în cadrul unui proces copil separat, toate create de către același părinte. TOATE PROCESURILE CREATE AR TREBUI SĂ RULEZE ÎN PARALEL. La sfârșitul fiecărui proces copil, părintele va prelua starea acestuia și va afișa un mesaj în următorul format:

```
Procesul cu PID-ul <PID> s-a încheiat cu codul <cod>
```

unde <PID> reprezintă PID-ul procesului copil terminat, iar <cod> reprezintă codul de ieșire cu care s-a încheiat.
Exemplu de cum ar trebui să funcționeze programul:

```
./program_exe -o director_iesire dir1 dir2 dir3
```

```
                        +------------------+
                        |  Proces Părinte  |
                        +------------------+
                                 |
                                 |
                +----------------+-------------------+
                |                |                   |
       +----------------+   +----------------+   +-----------------+
       | Proces Copil 1 |   | Proces Copil 2 |   | Proces Copil 3 |
       +----------------+   +----------------+   +-----------------+
       | Instantaneu    |   | Instantaneu    |   | Instantaneu     |
       | Creat          |   | Creat          |   | Creat           |
       | pentru dir1    |   | pentru dir2    |   | pentru dir3     |
       +----------------+   +--------------- +   +-----------------+


+------------------+     +--------------------------+    +--------------------------+     +---------------------
|  Proces Părinte  |---->| Închide Procesul Copil 1 |----| Închide Procesul Copil 2 |----| Închide Procesul Copi
+------------------+     +--------------------------+    +--------------------------+     +---------------------
```

Presupunând că programul afișează un mesaj după ce fiecare proces de creare a snapshot-ului este finalizat, ieșirea ar putea arăta asa:

```
Snapshot for Directory 1 created successfully.
Snapshot for Directory 2 created successfully.
Snapshot for Directory 3 created successfully.

Child Process 1 terminated with PID 123 and exit code 0.
Child Process 2 terminated with PID 124 and exit code 0.
Child Process 3 terminated with PID 125 and exit code 0.
```

Fiecare mesaj indică faptul că procesul de creare a snapshot-ului pentru fiecare director a fost finalizat fără erori.

Proiect SO Saptamana 9

În această săptămână, ne vom concentra pe analiza fișierelor dintr-un director primit ca argument în linia de comandă pentru a identifica și izola fișierele potențial periculoase sau corupte. Atacatorii utilizează mai multe tehnici, inclusiv criptarea șirurilor și apelurilor API, adăugarea de caractere aleatorii pentru a ascunde informația, includerea unor path-uri de fisiere executabile, avand denumiri explicite etc.

Pentru fiecare fișier găsit în directorul specificat care are toate drepturile lipsă, indicând astfel posibile semne de corupție sau maleficență, se va crea un proces nou care vom efectua o analiză sintactică a conținutului lui, pentru a preveni riscul de expunere la acțiuni daunatoare sistemului de operare.

Proiectul va fi structurat astfel:

1. **Verificarea Drepturilor Lipse**: Se va verifica dacă fișierul respectiv are toate drepturile lipsă, iar in cazul in care acest lucru se dovedeste sa fie adevarat, se va crea un proces dedicat care va realiza printr-un script (ex: `verify_for_malicious.sh`) o analiza sintactica, pentru a determina daca este corupt sau malitios, iar mai apoi ar trebui izolat intr-un mediu separat si safe.
2. **Analiză Sintactică a Fișierului**: În loc să deschidem direct fișierul, vom efectua o **analiză sintactică** a conținutului său pentru a identifica semne de maleficență sau corupție. Scriptul va include verificarea numărului de linii, cuvinte și caractere din fișier, precum și căutarea de cuvinte cheie asociate cu fișierele corupte sau malitioase, cum ar fi "corrupted", "dangerous", "risk", "attack", "malware", "malicious" sau chiar caractere non-ASCII. Daca se vor gasi una din aceste elemente, fișierul va fi considerat periculos și va fi izolat de restul fișierelor.
3. **Izolarea Fișierelor Periculoase**: Fișierele identificate ca periculoase vor fi mutate într-un director special, specificat ca argument în linia de comandă, numit "izolated_space_dir". Această acțiune va preveni expunerea la potențiale amenințări și va permite investigarea ulterioară a fișierelor suspecte.

Apelarea programului:

```
./program_exe -o director_iesire izolated_space_dir dir1 dir2 dir3
```

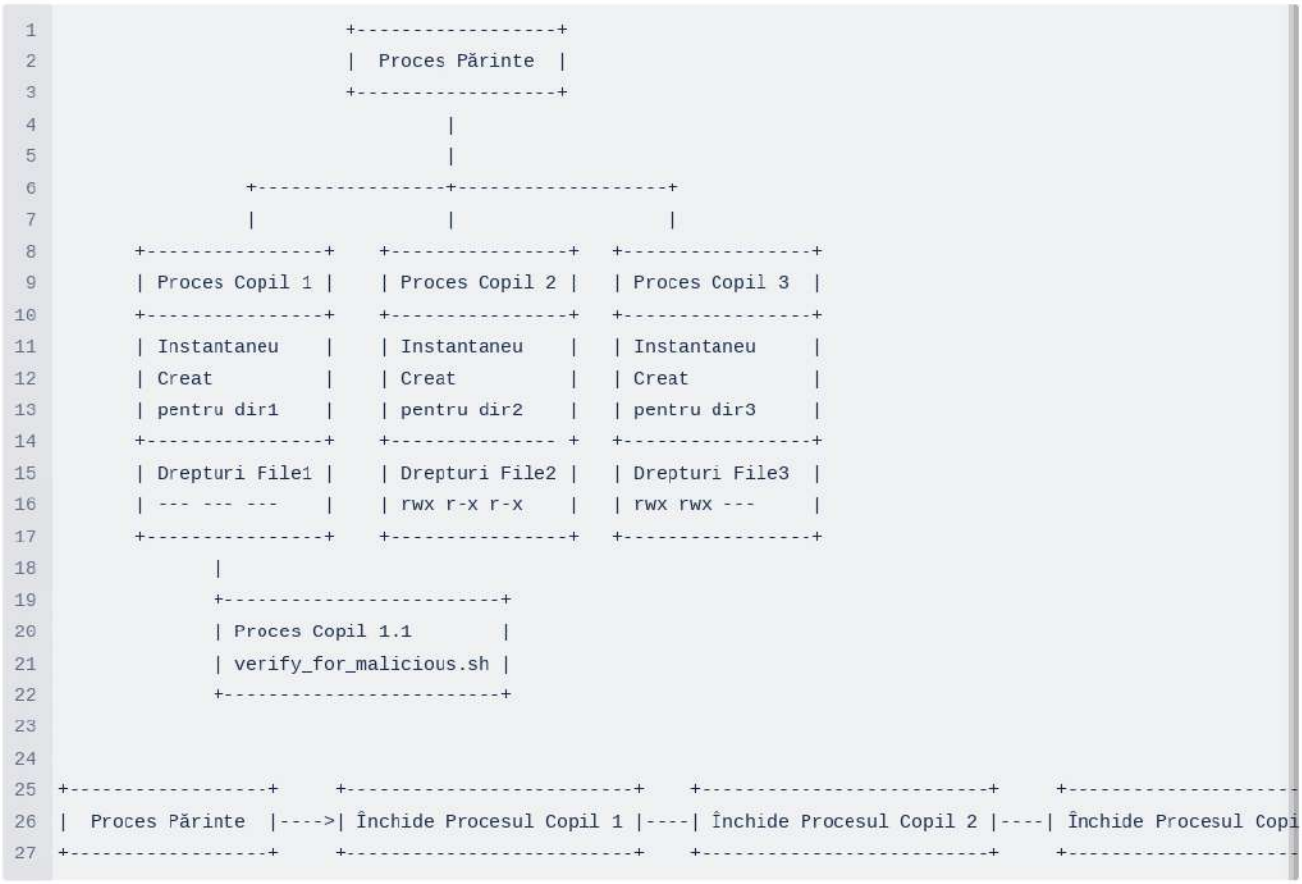
4. Pentru a îmbunătăți înțelegerea argumentelor de intrare ale programului, un nou indicator, "-s" (de la safe), va fi introdus ca argument în linia de comandă chiar înainte de argumentul "izolated_space_dir". Acesta va indica faptul că următorul argument este directorul desemnat pentru izolarea fișierelor potential periculoase. **Nu trebuie să existe un snapshot dedicat pentru acest director.** Programul va fi apelat:

```
./program_exe -o output_dir -s izolated_space_dir dir1 dir2 dir3
```

Prin aplicarea acestor măsuri, ne propunem să identificăm și să izolăm fișierele potențial periculoase sau corupte din directorul specificat, protejând astfel sistemul și datele împotriva amenințărilor de securitate.

Exemplu de structura a directoarelor:

```
Main Directory
|_ Dir1
|  |_ File1 (Drepturi: --- --- ---)
|_ Dir2
|  |_ File2 (Drepturi: rwx r-x r-x)
|_ File4.txt
|_ Dir3
|  |_ File3 (Drepturi: rwx rwx ---)
```



Proiect SO - Saptamana 10

Un fișier corupt poate să conțină o succesiune de caractere fără să respecte o structură de linii propriu-zisă. De asemenea, numărul de așa-numitele cuvinte poate fi redus, deoarece această succesiune de caractere poate fi fragmentată în moduri neașteptate. Pentru a evalua natura coruperii unui fișier, scriptul va număra câte linii, cuvinte și caractere conține acesta. Acest aspect poate reprezenta un filtru relevant pentru identificarea unui fișier periculos.

Exemplu de fisier corupt:

Pentru evaluarea fișierelor, următoarele criterii vor fi aplicate pentru **script-ul** realizat pentru analiza sintactica:

- **Numărul de linii, cuvinte și caractere:** Se vor stabili limite pentru fiecare dintre aceste aspecte.
- **Verificare a numărului de linii și cuvinte:** Dacă fișierul conține mai puțin de 3 linii **și** numărul de cuvinte depășește 1000 **și** numărul de caractere depășește 2000, va fi considerat suspect.
- **Verificare a caracterelor non-ASCII și cuvintelor cheie:** În cazul unui comportament suspect, se va examina dacă succesiunea de caractere conține și caractere non-ASCII **sau** cuvinte cheie asociate fișierelor periculoase ("corrupted", "dangerous", "risk", "attack", "malware", "malicious").
- **Clasificare și afișare**: Dacă se detectează caracteristici periculoase, fișierul va fi clasificat ca fiind periculos, iar scriptul va afișa la stdout **doar** numele acestuia. In caz contrar, scriptul va afisa la stdout **numai** cuvantul "SAFE".

În această săptămână, continuăm analiza fișierelor din directorul primit ca argument pentru a identifica și izola fișierele potențial periculoase sau corupte. Fiecare fișier considerat suspect va fi supus unei verificări prin execuția script-ului dedicat. Scriptul va furniza informații relevante despre fișierul respectiv prin intermediul stdout.

Procesul fiu care a executat scriptul va comunica cu procesul părinte prin intermediul unui **pipe**, transmițând informațiile cum ar fi numele fișierului in cazult in care este periculos, sau cuvantul "SAFE" in caz contrar. Procesul părinte va prelua aceste informații și va **decide** ulterior mutarea fișierului în directorul izolat în funcție de rezultatele obținute.
Nota: daca logica de mutare in directorul izolat s-a efectuat anterior prin diferite metode, de aceasta data, numai procesul care primeste prin pipe numele fisierului va face aceasta mutare.

Putem întâlni multiple fișiere corupte în același director. Astfel, vom avea nevoie de pipe-uri pentru a comunica informațiile despre aceste fișierele corupte între procesele fiu și părinte. Fiecare fișier corupt va fi supus aceleiași proceduri de verificare și execuție a scriptului ca și în săptămâna precedentă, iar informațiile rezultate vor fi transmise prin pipe către procesul care l-a creat. Procesul creator va prelua aceste informații și va decide cum să gestioneze fiecare fișier corupt în consecință, inclusiv mutarea lor în directorul izolat.

```
Proces Părinte (P1)
|
+--- Proces Copil (P2) --------------------------+
|    |
|    +--- Proces Nepot (P2.1)  ----[pipe]---> P2 |
|    |
|    +--- Proces Nepot (P2.2)  ----[pipe]---> P2 |
|
|
+--- Proces Copil (P3) --------------------------+
|    |
|    +--- Proces Nepot (P3.1)  ----[pipe]---> P3 |
|    |
|    +--- Proces Nepot (P3.2)  ----[pipe]---> P3 |
|
|
+--- Proces Copil (P4) --------------------------+
     |
     +--- Proces Nepot (P4.1)  ----[pipe]---> P4 |
     |
     +--- Proces Nepot (P4.2)  ----[pipe]---> P4 |


```

La final, dorim ca procesul părinte să preia statusul copiilor săi și să afișeze un mesaj care să conțină acet statusul și numărul de fișiere corupte găsite.

```
Procesul Copil 1 s-a încheiat cu PID-ul 123 și cu 0 fisiere cu potential periculos.
Procesul Copil 2 s-a încheiat cu PID-ul 124 și cu 2 fisiere cu potential periculos.
Procesul Copil 3 s-a încheiat cu PID-ul 125 și cu 1 fisiere cu potential periculos.
```