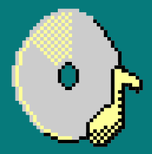


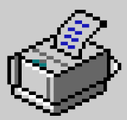





Encryptor

Pincu Iulia Maria Andreea

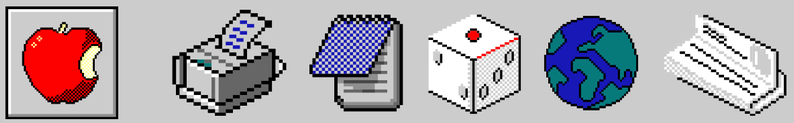
Universitatea din Bucuresti
Departamentul de Informatica
Sisteme de operare
2023-2024



11:11PM

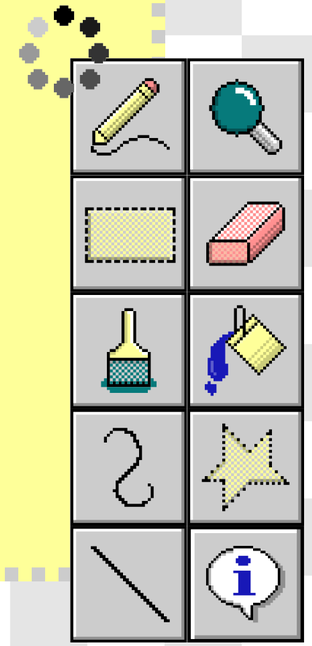
Cuprins



Back to Agenda Page

Descrierea problemei

Back to



Descrierea problemei

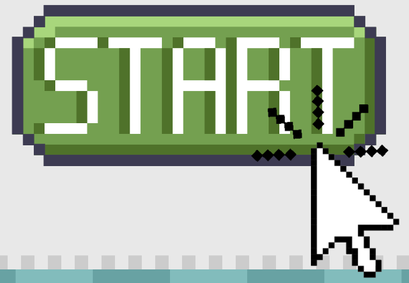
Securitatea datelor

În cadrul domeniului tehnologiei informației, unde securitatea datelor este de o importanță crucială, ne confruntăm cu o provocare semnificativă: optimizarea procesului de criptare și decriptare a textului. Această problemă complexă se încadrează într-un context general care implică alocarea eficientă a memoriei, planificarea meticuloasă a proceselor și sincronizarea precisă.

Criptare/Decriptare

- Particularitățile acestei probleme includ criptarea textului prin aplicarea permutărilor aleatoare pentru fiecare cuvânt și decriptarea acestuia prin utilizarea permutărilor specificate.
- Scopul este implementarea unei soluții eficiente care să optimizeze utilizarea resurselor de sistem în timpul criptării și decriptării textului.
- Importanța acestei probleme rezidă în creșterea securității informațiilor și îmbunătățirea performanței în procesarea textelor de dimensiuni mari.

Specificatia solutiei



Specificatia solutiei

Obiectivele

- Crearea unui sistem eficient și sigur pentru manipularea textului.
- Demonstrarea capacității de a gestiona procese multiple și memorie partajată.

Caracteristicile Prototipului

- Un sistem care criptează/decriptează textul, împărțind sarcina între mai multe procese.
- Interfață simplă de utilizare, cu un singur fișier de intrare pentru criptare si doua pentru decriptare.

Presupuneri și Constrângeri

- În funcție de numărul de argumente se realizează criptarea sau decriptarea.
- Limitări privind dimensiunea textului bazată pe capacitatea de memorie.

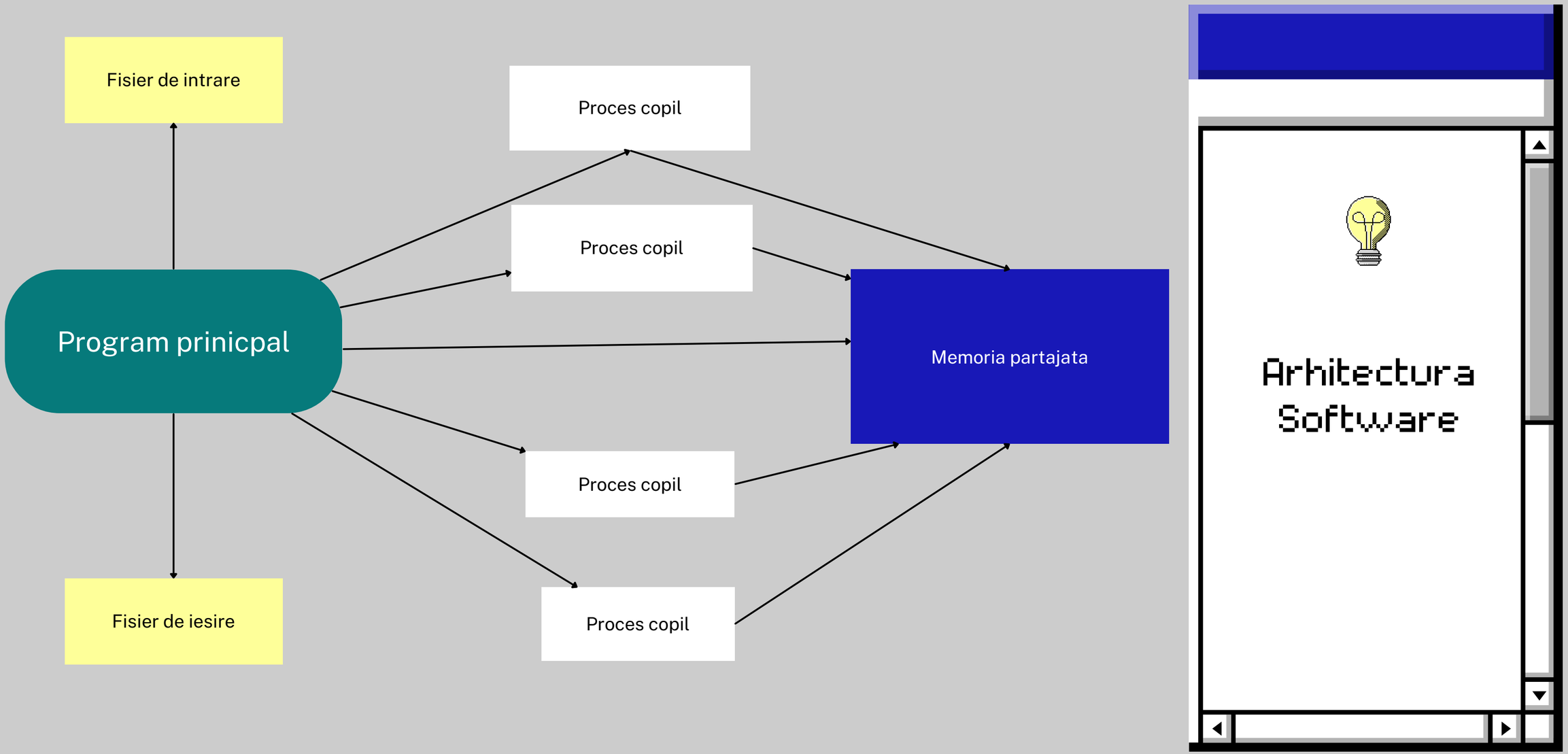
Cerințe și Specificații

- Sistem de operare compatibil cu procese multiple și memorie partajată.
- Biblioteci software pentru manipularea stringurilor și generarea numerelor aleatoare.
- Resurse minime necesare: memorie suficientă pentru stocarea textului și permutărilor.

Plan de Evaluare

- Testarea timpului de răspuns pentru diferite dimensiuni ale textului.
- Verificarea corectitudinii textului criptat/decriptat.


A
Design


Fisier de intrare
Proces copil
Proces copil
Program prinicpal
Memoria partajata
Proces copil
Proces copil
Fisier de iesire
Arhitectura
Software

Alegerea Mecanismelor:

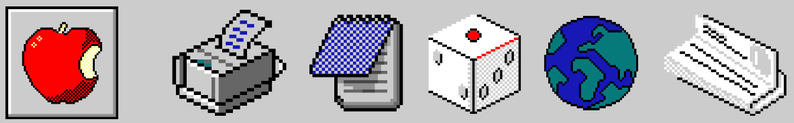
- Memoria partajată este aleasă pentru eficiența în comunicarea între procese și accesul rapid la date.
- Procesele multiple sunt utilizate pentru a îmbunătăți performanța prin paralelizare.

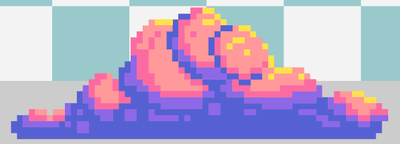
Descrierea Mecanismelor:

- Criptarea: Fiecare proces modifică un cuvânt folosind o permutare aleatoare.
- Decriptarea: Procesele folosesc permutările date pentru a reface textul original.

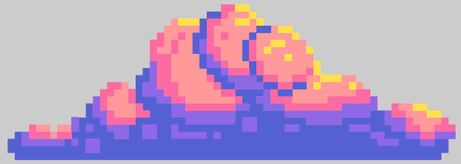
Particularități și Adaptări:

- Sistemul este adaptat pentru a gestiona texte de dimensiuni mari
- Mecanismele sunt optimizate pentru a minimiza timpul de așteptare și utilizarea resurselor.





Implementare



Descrierea Componentelor Software

- Funcție de Criptare/Decriptare: Algoritmul de amestecare pentru permutarea cuvintelor (shuffle_array) și logica de criptare și decriptare implementată în funcția main.
- Gestionarea Fișierelor și I/O: Codul deschide, citește, scrie și închide fișiere pentru a procesa textul sursă, fișierul criptat/decriptat și fișierul de permutări.
- Memorie Partajată: Utilizarea memoriei partajate pentru a stoca temporar cuvintele în timpul criptării/decriptării.

- Procese Multiple: Crearea de procese folosind fork() pentru a paraleliza criptarea/decriptarea.
- Gestionarea Erorilor: Verificarea și raportarea erorilor care pot apărea în timpul execuției programului (de exemplu, la deschiderea fișierelor sau la lucrul cu memoria partajată).

Biblioteci Software Utilizate

stdio.h, stdlib.h, string.h: Pentru operațiuni standard de intrare/ieșire, manipularea șirurilor de caractere și alocări de memorie.

sys/types.h, sys/stat.h, fcntl.h: Pentru lucrul cu fișiere și descriptori de fișiere.

sys/wait.h: Pentru gestionarea proceselor copil create cu fork().

sys/mman.h, sys/shm.h: Pentru utilizarea memoriei partajate.

unistd.h: Pentru apeluri de sistem ca fork(), getpagesize() și alte funcții specifice sistemului POSIX.

errno.h: Pentru gestionarea și raportarea erorilor.

time.h: Pentru generarea unor numere aleatoare bazate pe timp în funcția shuffle_array.

Probleme Tehnice Întâmpinate

Gestionarea Memoriei Partajate

- Se realizează folosind funcții precum shm_open(), ftruncate(), mmap(), și munmap()
- Gestionarea necorespunzătoare a accesului la memorie partajată între procese poate duce la probleme de acces concurent și coruperea datelor partajate.
- Dacă dimensiunea memoriei partajate nu este calculată corespunzător, s-ar putea să nu fie suficient spațiu pentru datele care trebuie stocate, sau s-ar putea aloca prea multă memorie inutil.

Sincronizarea Proceselor

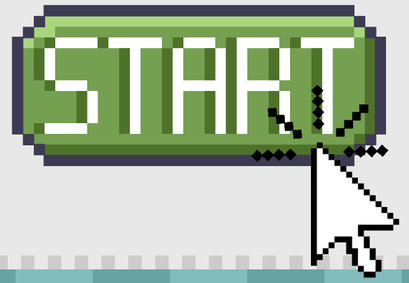
Utilizarea fork(): Programul creează un proces copil pentru fiecare cuvânt care trebuie criptat/decriptat.
Rolul wait(NULL): După fiecare apel fork(), programul apelează wait(NULL) în procesul părinte. Aceasta face ca procesul părinte să aștepte finalizarea procesului copil înainte de a continua execuția. Funcția wait() blochează procesul părinte până când unul dintre procesele sale copil se termină.

Gestionarea erorilor

Am gestionat următoarele feluri de erori:

- erorile care ar fi putut apărea în lucrul cu fișiere (dacă nu se pot deschide fișierele, se afișează un mesaj de eroare utilizatorului cu ajutorul funcției perror())
- erori de alocare a memoriei partajate (codul verifică dacă crearea și gestionarea memoriei partajate au fost efectuate cu succes folosind funcția shm_open(). În cazul unei erori, se afișează un mesaj de eroare utilizatorului cu ajutorul funcției perror())
- erori la crearea proceselor copil

Experimente



Descrierea arhitecturii

- Proiectul a fost realizat pe o mașină virtuală cu Ubuntu 22.04.3 LTS
- Caracteristicile HW pot fi observate in această poză sau utilizând comanda lscpu în terminal

Use Case-uri

- Am folosit tool-ul Valgrind pentru identificarea problemelor legate de memorie, cum ar fi scurgeri de memorie (memory leaks), utilizarea memoriei neinițializate sau alte erori legate de gestionarea memoriei.
- Am folosit comenda time pentru a vizualiza durata de rulare a programului și pentru a evalua performanța programului

Criptare

0 cuvinte

Decriptare

```
pinku@pinku-VirtualBox:~/SO/Project$ time ./encryptor zero_words
File successfully encrypted!

real    0m0,002s
user    0m0,000s
sys     0m0,002s
pinku@pinku-VirtualBox:~/SO/Project$ time ./encryptor zero_wordsEncrypted.txt pe
rmutations
File successfully decrypted!

real    0m0,002s
user    0m0,002s
sys     0m0,000s
pinku@pinku-VirtualBox:~/SO/Project$ S
```

Criptare

1000 cuvinte

Decriptare

```
pinku@pinku-VirtualBox:~/SO/Project$ time ./encryptor 1000_words
File successfully encrypted!

real    0m1,163s
user    0m0,424s
sys     0m0,742s
pinku@pinku-VirtualBox:~/SO/Project$ time ./encryptor 1000_wordsEncrypted.txt pe
rmutations
File successfully decrypted!

real    0m1,136s
user    0m0,398s
sys     0m0,746s
```

Criptare

10000 cuvinte

Decriptare

```
pinku@pinku-VirtualBox:~/SO/Project$ time ./encryptor 10000_words
File successfully encrypted!

real    1m10,621s
user    0m0,910s
sys     1m9,759s
pinku@pinku-VirtualBox:~/SO/Project$ time ./encryptor 10000_wordsEncrypted.txt p
ermutations
File successfully decrypted!

real    1m10,421s
user    0m1,089s
sys     1m9,419s
pinku@pinku-VirtualBox:~/SO/Project$
```

Concluzii

Am învățat despre:

- gestionarea emoriei partajate
- utilizarea proceselor
- maparea fișierelor
- sincronizarea proceselor
- importanța criptării
- utilizarea corectă a fișierelor