**Proiectarea Sistemelor de Operare**

-C113E-

<GsK FileSystem – Sistem de fisiere Unix simplificat >

*Sd. Sg. Gheorghe Stefan-Lucian*

*Sd. Sg. Radulescu Silviu-Sebastian*

# Prezentare Generala

GsK FileSystem este o implementare simplificata a sistemului de fisiere Unix. Acesta ofera o aplicatie shell de baza pentru interactiunea cu un sistem de fisiere, permitand operatiuni fundamentale precum crearea, stergerea, citirea si scrierea fisierelor. Proiectul este impartit in trei componente principale:

1. **Shell:**

O aplicatie de comanda simpla care traduce comenzi ale utilizatorului in operatiuni ale sistemului de fisiere

1. **Sistem de fisiere:**

Se ocupa cu organizarea structurilor de date pe disc si efectueaza toata evidenta necesara pentru stocarea a datelor.

1. **Emulator disc:**

Emuleaza un disc impartind un fisier normal ( numit imagine de disc ) in blocuri de 4096 de octeti si permite sistemului de fisiere sa citeasca si sa scrie in termeni de blocuri.

# Instalare si utilizare

# Structura proiectului

Proiectul contine mai multe fisiere sursa esentiale:

1. “disk.h” si “disk.cpp”: Definesc si implementeaza emulatorul de disc;
2. “fs.h” si “fs.cpp”: Definesc si implementeaza logica sistemului de fisiere;
3. “sfssh.cpp”: Implementeaza shell-ul prin care se interactioneaza cu sistemul de fisiere.

# Componentele sistemului de fisiere

1. **SuperBlock:**

SuperBlock-ul este primul bloc de pe disc si are un rol crucial in sistemul de fisiere. El contine metadata care descriu structura si dimensiunile sistemului de fisiere. Acesta include:

* **Magic Number:** o semnatura specifica sistemului de fisiere care este verificata la montarea discului pentru a valida integritatea sistemului de fisiere;
* **Blocks:** numarul total de blocuri de pe disc;
* **InodesBlocks:** numarul total de blocuri alocate pentru inodes, de obicei 10% din numarul total de blocuri;
* **Inodes:** numarul total de inodes care pot fi stocate pe disc.

1. **Inode:**

Inode-urile sunt structure de date care reprezinta fisiere si directoare. Fiecare inode include:

* **Valid:** un indicator care arata daca inode-ul este in uz sau nu;
* **Size:** dimensiunea fisierului in octeti;
* **Direct Pointers:** pana la 5 pointeri catre blocuri de date care stocheaza continutul fisierului;
* **Indirect Pointer:** un pointer catre un bloc de date care contine la randul sau pointeri catre alte blocuri de date, extinzand astfel capacitatea de stocare a unui fisier.

1. **Blocuri de date:**

Blocurile de date sunt segmente de 4096 de octeti care stocheaza efectiv datele utilizatorului. Blocurile pot fi alocate ca blocuri directe sau indirecte prin inode-uri.

1. **Bitmap-ul de blocuri libere:**

Bitmap-ul este o structura de date in memorie, nu pe disc, care urmareste care blocuri sunt libere sau ocupate. Cand sistemul de fisiere are nevoie de un bloc nou, verifica bitmap-ul pentru a gasi un bloc liber. Cand un bloc nu mai e necesar, este marcat ca liber in bitmap.

1. **Blocuri indirecte:**

Blocurile indirecte sunt folosite cand un fisier depaseste dimensiunea ce poate fi indexata direct prin inode. Fiecare bloc indirect contine un set de pointeri catre alte blocuri de date, permitand extinderea spatiului disponibil pentru un singur fisier.

1. **Recuperarea bitmap-ului de blocuri libere:**

La fiecare montare a sistemului de fisiere este necesara reconstructia bitmap-ului de blocuri libere prin scanarea tuturor inode-urilor pentru a determina care blocuri sunt in uz.

1. **Scalabilitate si limitari:**

Desi GsK FileSystem este un sistem de fisiere functional, este limitat de simplificarea sa. De exemplu nu suporta nomenclatura fisierelor si a directoarelor, iar simplificarea sa il face sa nu fie optimizat pentru performante in productie.

# Comenzi disponibile in shell

printf(" format\n"); printf(" mount\n")printf(" unmount\n"); printf(" debug\n"); printf(" create\n") printf(" remove <inode>\n"); printf(" stat <inode>\n") printf(" write <inode> <text>\n"); printf(" cat <inode>\n"); printf(" help\n"); printf(" quit\n"); printf(" exit\n");

* **Help – “help”:** afisarea comenzilor disponibile in shell;
* **Format – “format”:** initializeaza discul cu un nou sistem de fisiere, creand un superblock si pregatind blocurile pentru inodes si date. Acesta va sterge toate datele existente pe disc;
* **Mount – “mount”:** monteaza discul pentru a putea fi utilizat, incarcand superblock-ul si reconstruind bitmap-ul de blocuri libere din memoria sistemului de fisiere;
* **Debug – “debug”:** afiseaza informatii de depanare despre structura sistemului de fisiere, inclusive superblock-ul si starea inode-urilor;
* **Create – “create”:** creeaza un nou inode( fisier ) in sistemul de fisiere si returneaza numarul inode-ului creat;
* **Remove – “remove <inode>”:** sterge fisierul asociat inode-ului dat, eliberand blocurile de date asociate si marcand inode-ul ca nevalid!!!!!;
* **Stat – “stat <inode>”:** afiseaza informatii despre fisierul asociat inode-ului, inclusive marimea fisierului si blocurile de date la care pointeaza;
* **Write – “write <inode> <text>”:** scrie textul furnizat in blocurile de date asociate cu inode-ul specificat, incepand de la primul bloc direct si continuand in blocuri suplimentare daca este necesar;
* **Cat – “cat <inode>”:** citeste si afiseaza continutul fisierului asociat cu inode-ul dat, folosind blocuri de date directe si indirecte daca este cazul;
* **Exit – “exit”:** inchide shell-ul si demonteaza sistemul de fisiere, operatiile fiind finalizate si datele ramanand pe disc.

# Bibliografie

* Project Simple File System: <https://www3.nd.edu/~pbui/teaching/cse.30341.fa19/project06.html>
* File System Implementation: <https://pages.cs.wisc.edu/~remzi/OSTEP/file-implementation.pdf>