DESENVOLVIMENTO DE UM SISTEMA DE ARQUIVOS INSTALÁVEL PARA LINUX

Thiago Klein Flach

Orientador: Mauro Marcelo Mattos

ROTEIRO

1 – Introdução:

1.1 – Objetivos.

2 – Fundamentação Teórica:

2.1 – Conceitos Básicos;

2.2 – Contexto Atual.

3 – Desenvolvimento:

3.1 – Requisitos Principais;

3.2 – Especificação;

3.3 – Implementação;

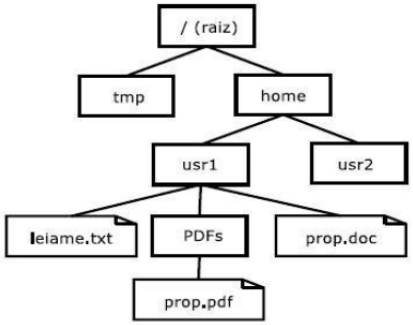
3.4 – Resultados e Discussão.

4 – Conclusão:

4.1 – Extensões.

1 - Introdução

- **Sistema de Arquivos:**
  - Camadas (interface | dados e drivers);
  - Interface entre o usuário e os dados.



- **Virtual File System (VFS):**
  - Interface comum e transparente, independente do sistema de arquivos utilizado.

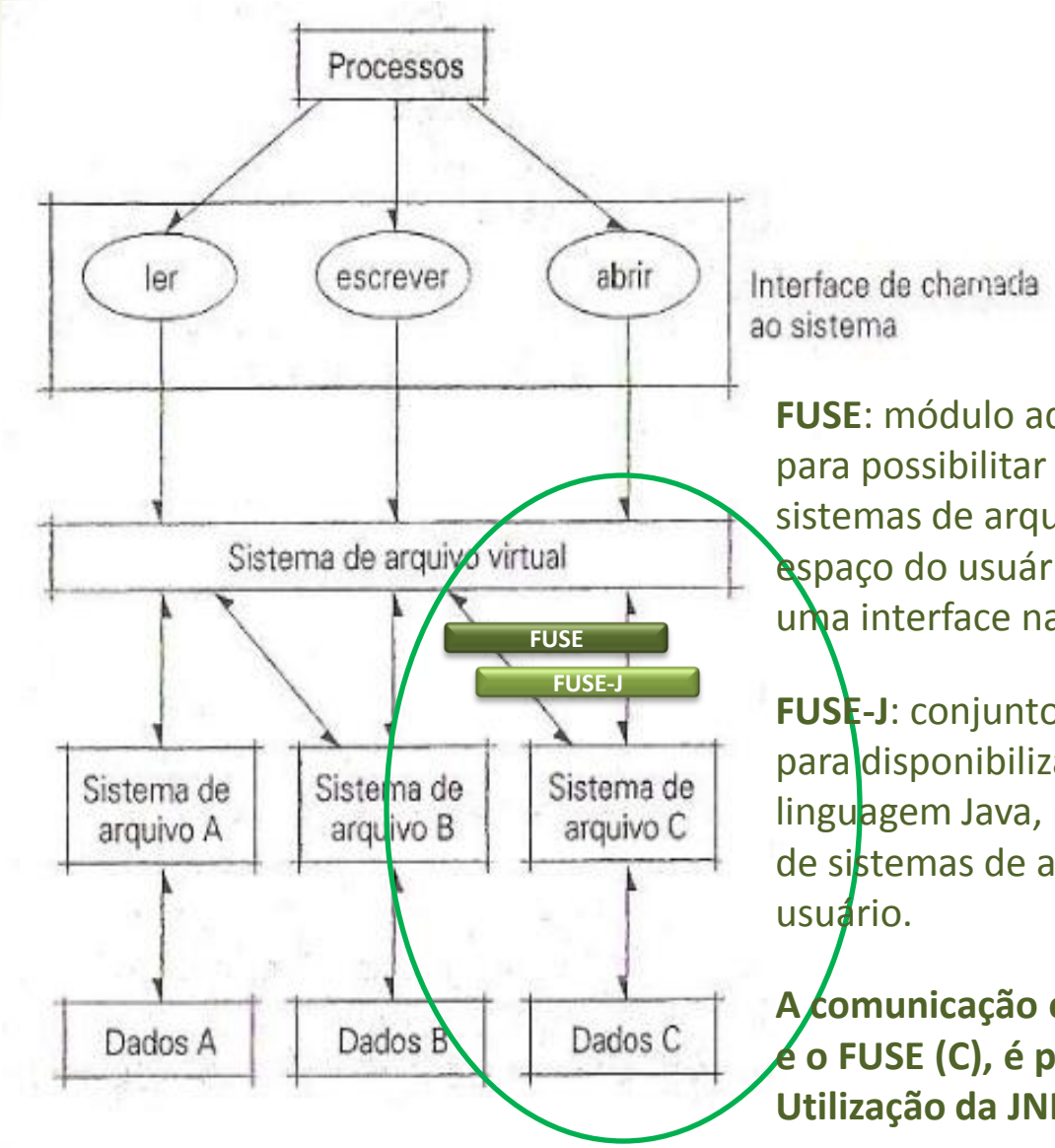
- **Desenvolvimento:**
  - Complexidade e abstração;
  - API (drivers e plataformas) para suportar recursos;
  - Sistema de arquivos instalável e com dispositivo específico.

1.1 - Objetivos

Desenvolvimento de um sistema de arquivos instalável em Linux (dispositivo específico).

- Estudo e descrição de aspectos estruturais;
- Desenvolvimento de um protótipo de validação.

2 - Fundamentação Teórica



FUSE: módulo adicionado ao Kernel para possibilitar a comunicação entre sistemas de arquivos executados no espaço do usuário e o VFS, através de uma interface na linguagem C.

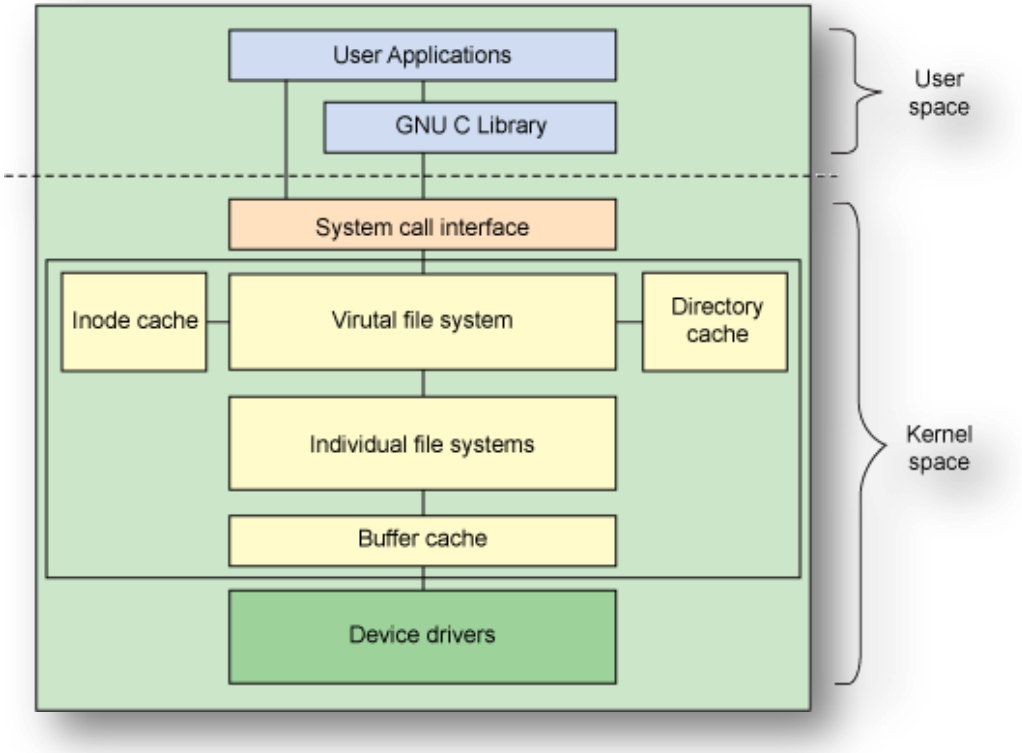
FUSE-J: conjunto de bibliotecas criadas para disponibilizar uma interface na linguagem Java, para o desenvolvimento de sistemas de arquivos do espaço do usuário.

A comunicação entre o FUSE-J (Java) e o FUSE (C), é possível por causa da Utilização da JNI (Java Native Interface).

2.1 – Conceitos Básicos

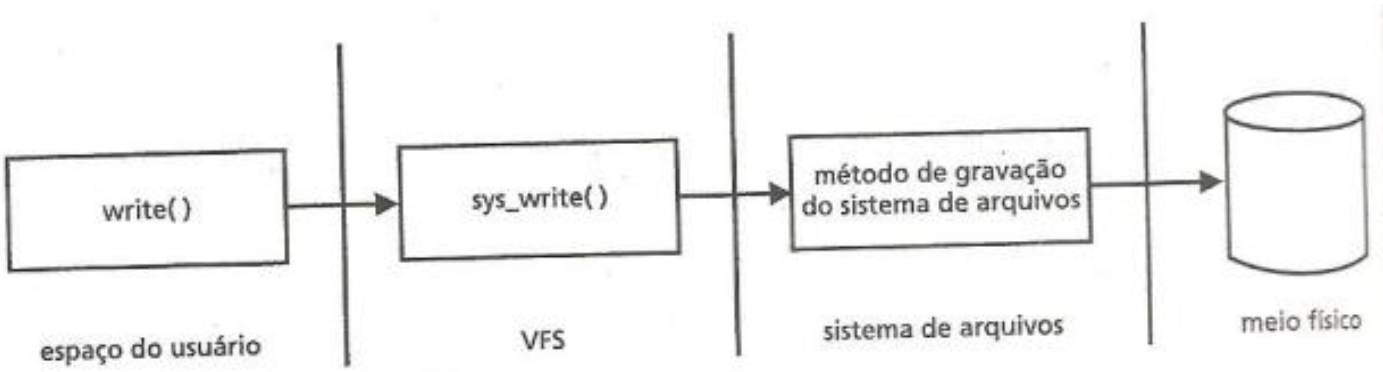
2.1.1 – SISTEMAS DE ARQUIVOS:

- Dados e metadados (*superblock*, *inode*);
- **Espaço do Usuário** X **Espaço do *Kernel***:



2.1.2 – VIRTUAL FILE SYSTEM (VFS):

- Estruturas de dados comuns entre os sistemas de arquivos;
- Um usuário pode acessar qualquer dado sem saber em qual dispositivo ou sistema de arquivos ele está.



– Permite o uso de chamadas padrão do sistema operacional para operar em diferentes sistemas de arquivos e meios:

FUNÇÃO	DESCRIÇÃO
llseek	Atualiza o ponteiro do arquivo para o deslocamento do dado.
read	Lê os dados de um arquivo.
write	Escreve os dados em um arquivo.
readdir	Retorna o próximo diretório em uma listagem de diretórios.
open	Cria um novo objeto de arquivo e loga-o ao objeto inode correspondente.
release	Esta função é chamada pelo VFS quando a última referência restante para o arquivo é destruída.

2.1.3 – SISTEMAS DE ARQUIVO NO ESPAÇO DO USUÁRIO:

Definição: sistemas de arquivos executados no ambiente de trabalho do usuário, podendo fazer uso dos recursos que ali se encontram, como: *variáveis*, *bibliotecas*, *plataformas*, *aplicações*, *rede*, *entre outros*.

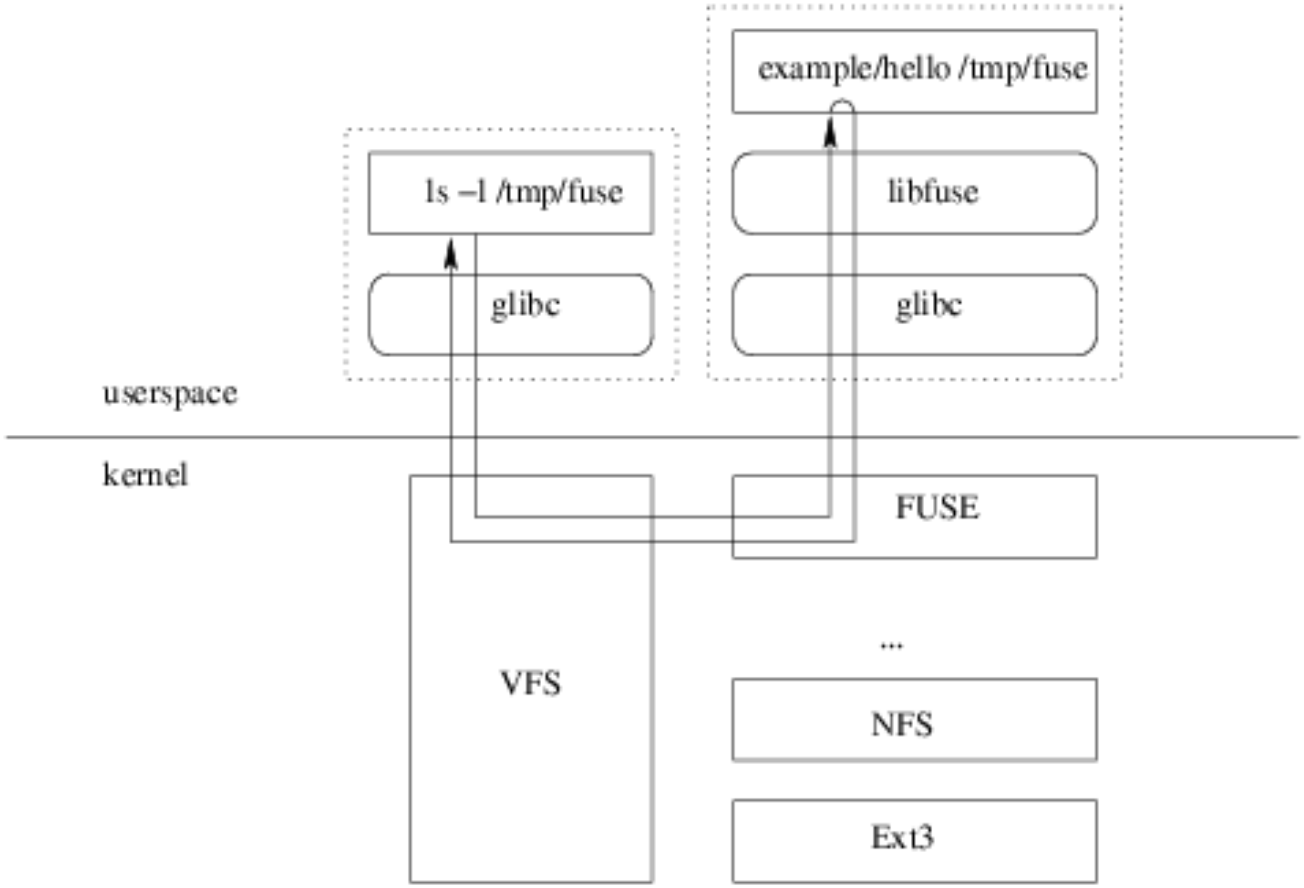
- **File System in User Space (FUSE)**: módulo que possibilita o desenvolvimento de sistemas de arquivos na linguagem C, executando-os no espaço do usuário.

- **File System in User Space Java API (FUSE-J)**: interface que possibilita do desenvolvimento de sistemas de arquivos na linguagem Java, executando-os sobre a *Java Virtual Machine* (JVM) no espaço do usuário.

2.1.3.1 – FILE SYSTEM IN USER SPACE (FUSE):

- Módulo instalável;
- Possibilita o desenvolvimento de sistemas de arquivos na linguagem C;
- Pode ser utilizado por usuários sem privilégio;
- Compatível com as versões 2.4 e 2.6 do Kernel;
- Baseia seus modelos estruturais no VFS;
- **Resolve as seguintes dificuldades comuns na implementação de sistemas de arquivos:**
  - **APIs extensas e complexas;**
  - **Trabalhar no espaço do Kernel dificulta depuração e potencializa o aparecimento de falhas críticas.**

– Fluxos:


example/hello /tmp/fuse
ls –l /tmp/fuse
libfuse
glibc
glibc
userspace
kernel
FUSE
VFS
...
NFS
Ext3

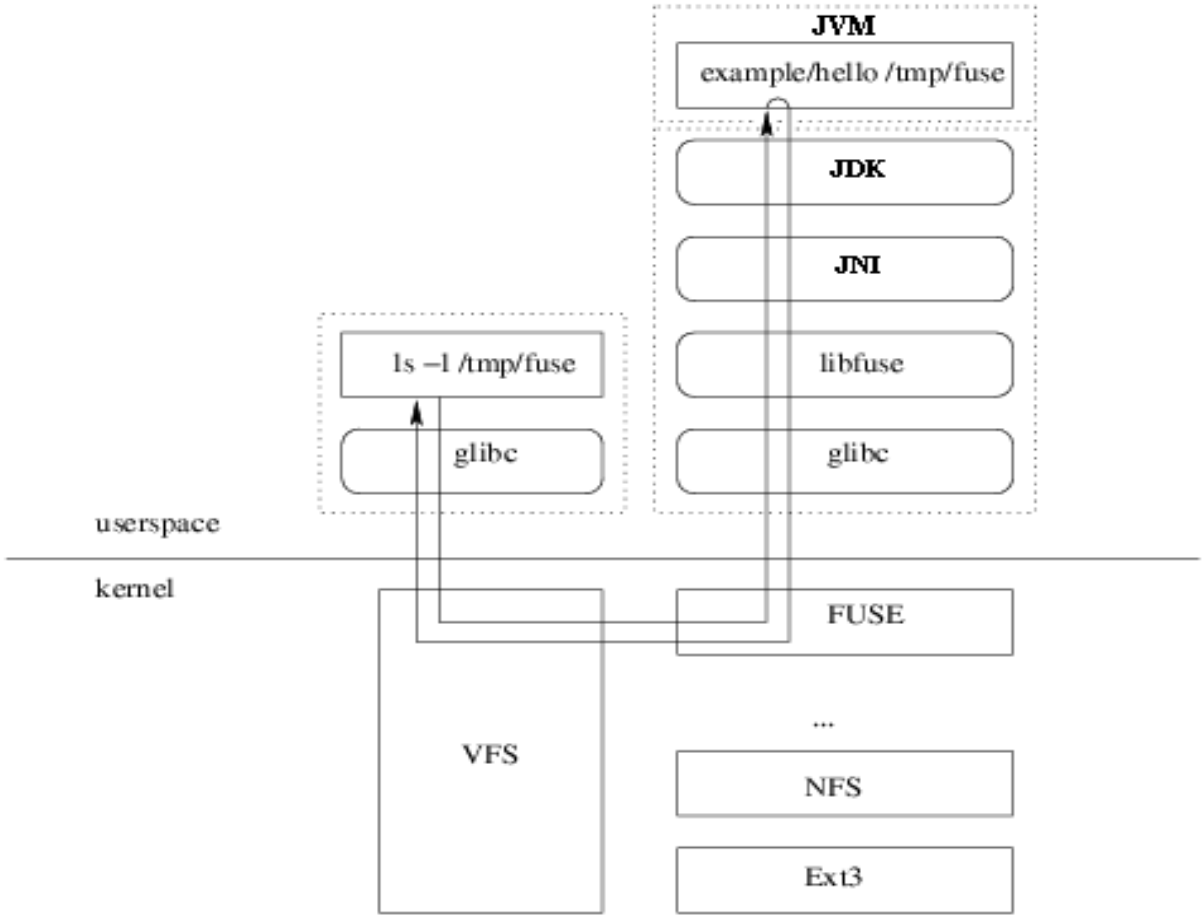
– As funções do FUSE são mais simplificadas quando comparadas às do VFS, pois os parâmetros recebidos são mais simples de manipular:

FUNÇÕES DO FUSE
getattr(name : String, stat : Stat)
readlink(name : String, buffer : char *, len : nativeint)
mknod(name : String, mode : nativeint, dev : nativeint)
mkdir(name : String, mode : nativeint)
open(name : String, info)
read(name : String, buffer : char *, size : nativeint, off : nativeint)
write(name : String, buffer : char *, size : nativeint, off : nativeint)
rename(name1 : String, name2 : String)
link(name1 : String, name2 : String)
chmod(name : String, mode : nativeint)
chown(name : String, uid : nativeint, gid : nativeint)
truncate(name : String, len : nativeint)

```c
#include <fuse.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <fcntl.h>

static int hello_getattr(const char *path, struct stat *stbuf)
{
    (...)
}

static int hello_readdir(const char *path, void *buf, fuse_fill_dir_t filler,
                         off_t offset, struct fuse_file_info *fi)
{
    (...)
}

static int hello_open(const char *path, struct fuse_file_info *fi)
{
    (...)
}

static int hello_read(const char *path, char *buf, size_t size, off_t offset,
                      struct fuse_file_info *fi)
{
    (...)
}

static struct fuse_operations hello_oper = {
    .getattr        = hello_getattr,
    .readdir        = hello_readdir,
    .open = hello_open,
    .read = hello_read,
};

int main(int argc, char *argv[])
{
    return fuse_main(argc, argv, &hello_oper);
}
```

2.1.3.2 – FILE SYSTEM IN USER SPACE JAVA API (FUSE-J):

- API Java;
- Utiliza o JNI (*Java Native Interface*);
- Fluxo (FUSE x FUSE-J):



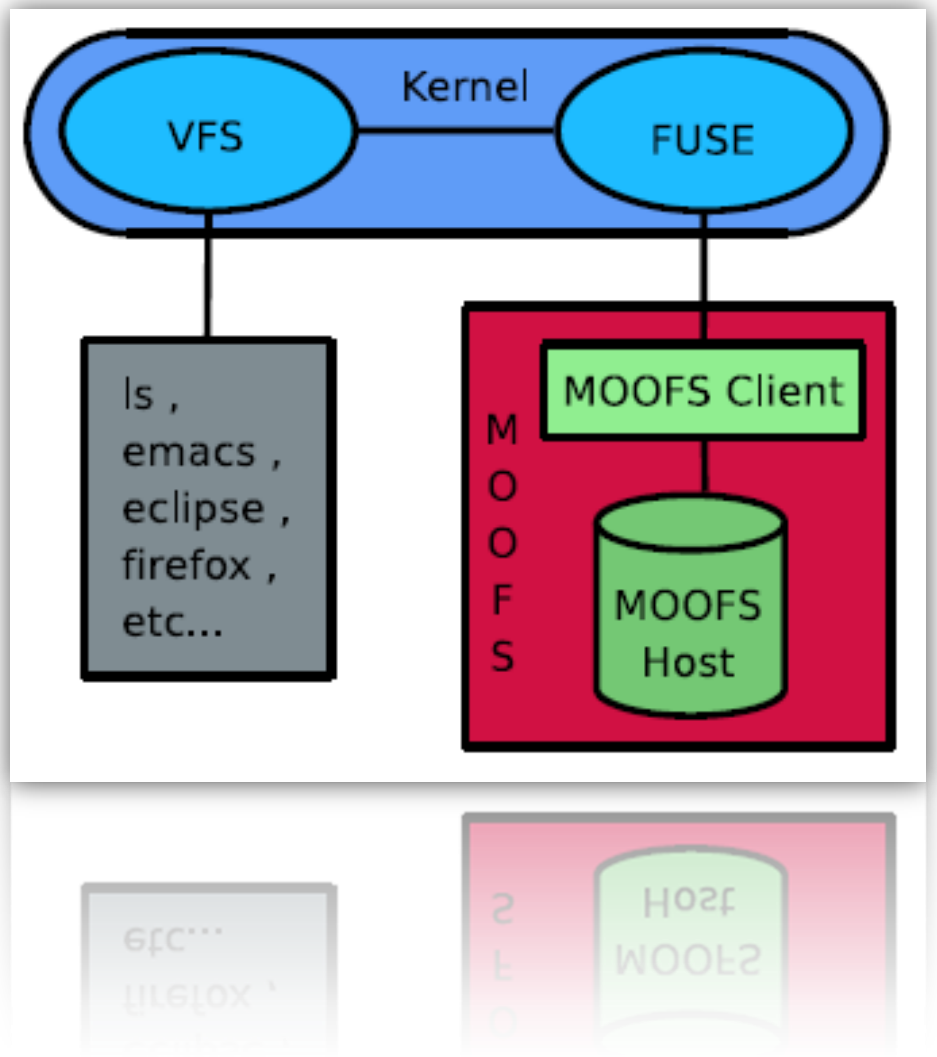
– Interface:

```
public interface Filesystem1 extends FilesystemConstants
{
   public FuseStat getattr(String path) throws FuseException;

   public String readlink(String path) throws FuseException;

   public FuseDirEnt[] getdir(String path) throws FuseException;

   public void mknod(String path, int mode, int rdev) throws FuseException;

   public void mkdir(String path, int mode) throws FuseException;

   public void unlink(String path) throws FuseException;

   public void rmdir(String path) throws FuseException;

   public void symlink(String from, String to) throws FuseException;

   public void rename(String from, String to) throws FuseException;

   public void link(String from, String to) throws FuseException;

   public void chmod(String path, int mode) throws FuseException;

   public void chown(String path, int uid, int gid) throws FuseException;

   public void truncate(String path, long size) throws FuseException;

   public void utime(String path, int atime, int mtime) throws FuseException;

   public FuseStatfs statfs() throws FuseException;

   public void open(String path, int flags) throws FuseException;

   public void read(String path, ByteBuffer buf, long offset) throws FuseException;

   public void write(String path, ByteBuffer buf, long offset) throws FuseException;

   public void release(String path, int flags) throws FuseException;
}
```

2.2 – Contexto Atual

2.2.1 – MOBILE OBJECTS ORIENTED FILE SYSTEM (MOOFS):

- Interface comum;
- Sistema de arquivos distribuído;
- Cada arquivo encontra-se espalhado em vários computadores;
- Uma máquina não armazena um arquivo inteiro, apenas blocos;
- Oportunista;
- Redundante;
- Cliente/servidor;
- Desenvolvido na linguagem: Phyton;
- **Executa sobre o FUSE.**

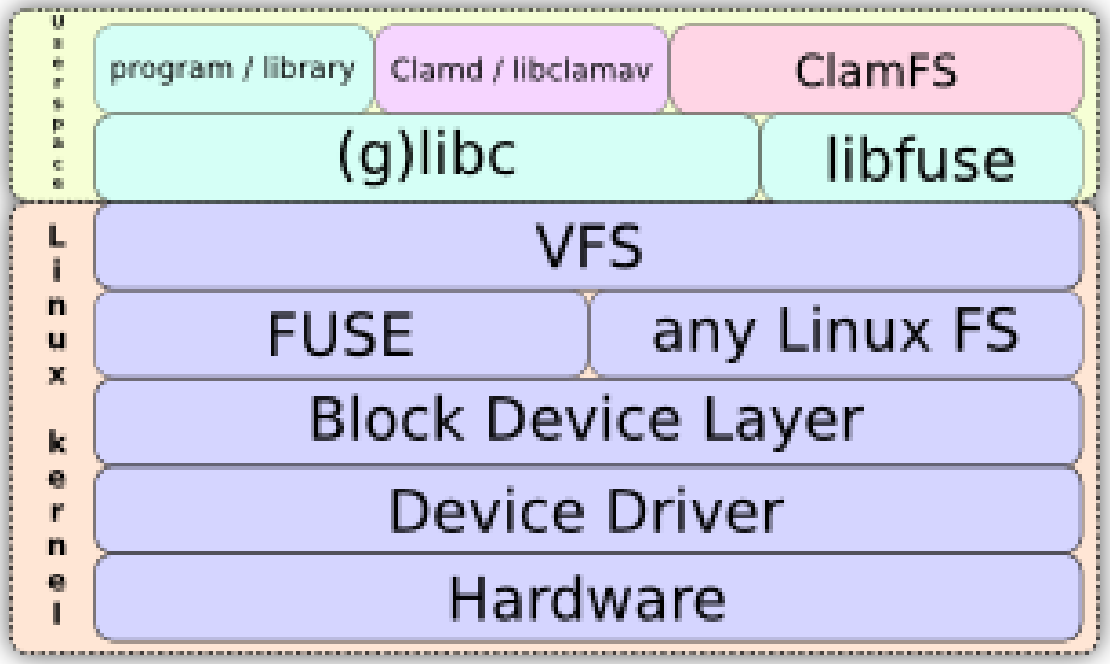

Kernel
VFS
FUSE
ls ,
emacs ,
eclipse ,
firefox ,
etc...
MOOFS Client
MOOFS
MOOFS
Host

2.2.2 – BLOGGER FILE SYSTEM (BloggerFS):

- Interface comum;
- Manipular postagens em blogs;
- Parâmetros:
  - Ponto de montagem;
  - Usuário;
  - Senha.
- **Desenvolvido com o FUSE-J;**
- **Executado sobre o FUSE.**

2.2.3 – CLAM FILE SYSTEM (ClamFS):

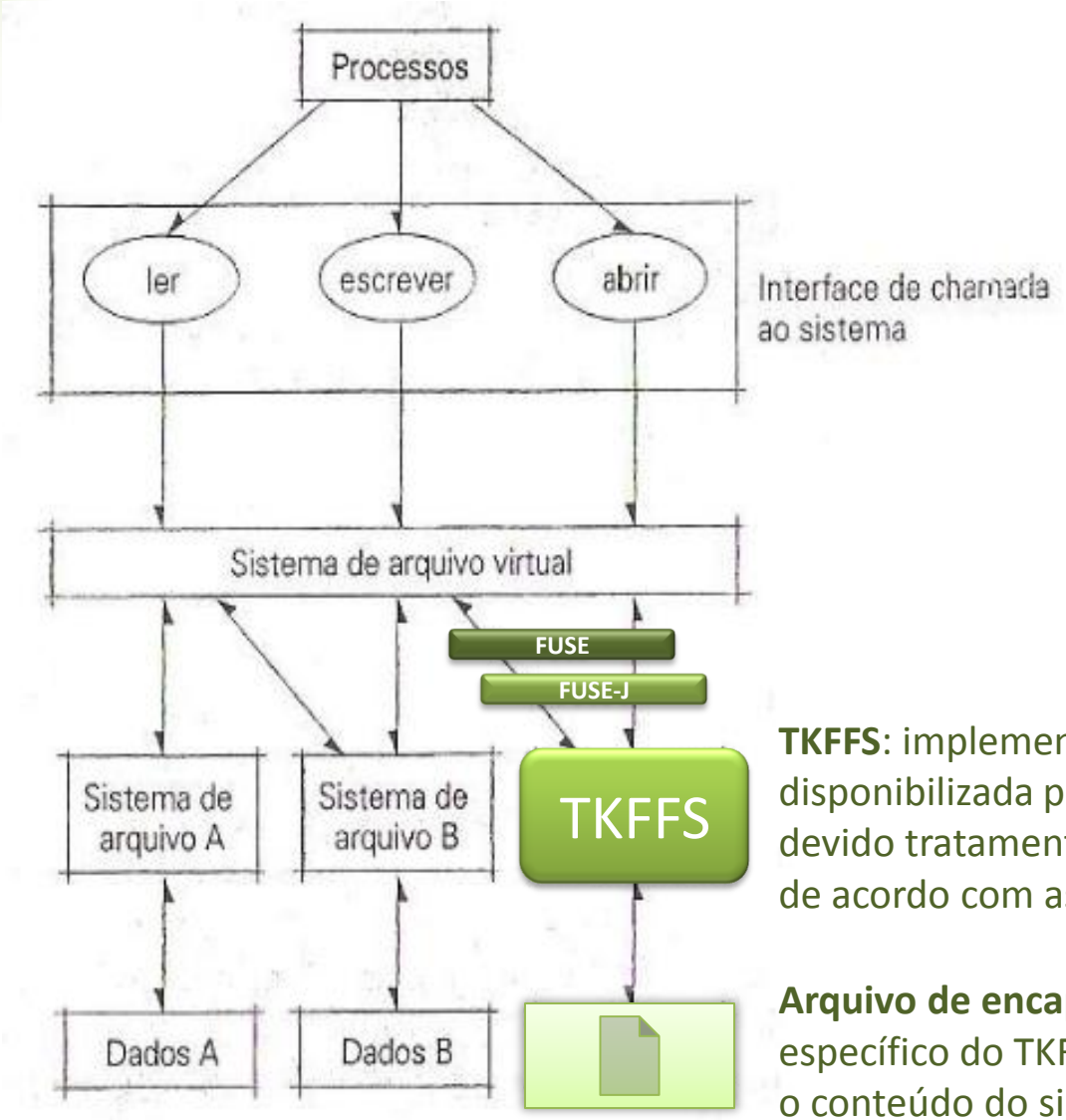
- Utiliza o Clam Anti Virus (ClamAV);
- O damon clamd efetua uma varredura nos arquivos que são manipulados;
- Envia um e-mail ao administrador caso um vírus for encontrado;
- **Executa sobre o FUSE.**



2.2.4 – QUESTÕES NÃO ESCLARECIDAS OU EM ABERTO:

- Armazenamento em dispositivos virtuais;
- Utilização do mesmo dispositivo virtual, com os mesmos dados nele contidos, em máquinas distintas;
- Realização de cópias de segurança dos dados do sistema de arquivos sem ter que copiá-los para outro sistema de arquivos.

3 - Desenvolvimento



TKFFS: implementa a interface disponibilizada pelo FUSE-J, dando o devido tratamento às operações, de acordo com as solicitações do FUSE.

Arquivo de encapsulamento: dispositivo específico do TKFFS, onde é armazenado o conteúdo do sistema de arquivos.

3.1. Requisitos principais

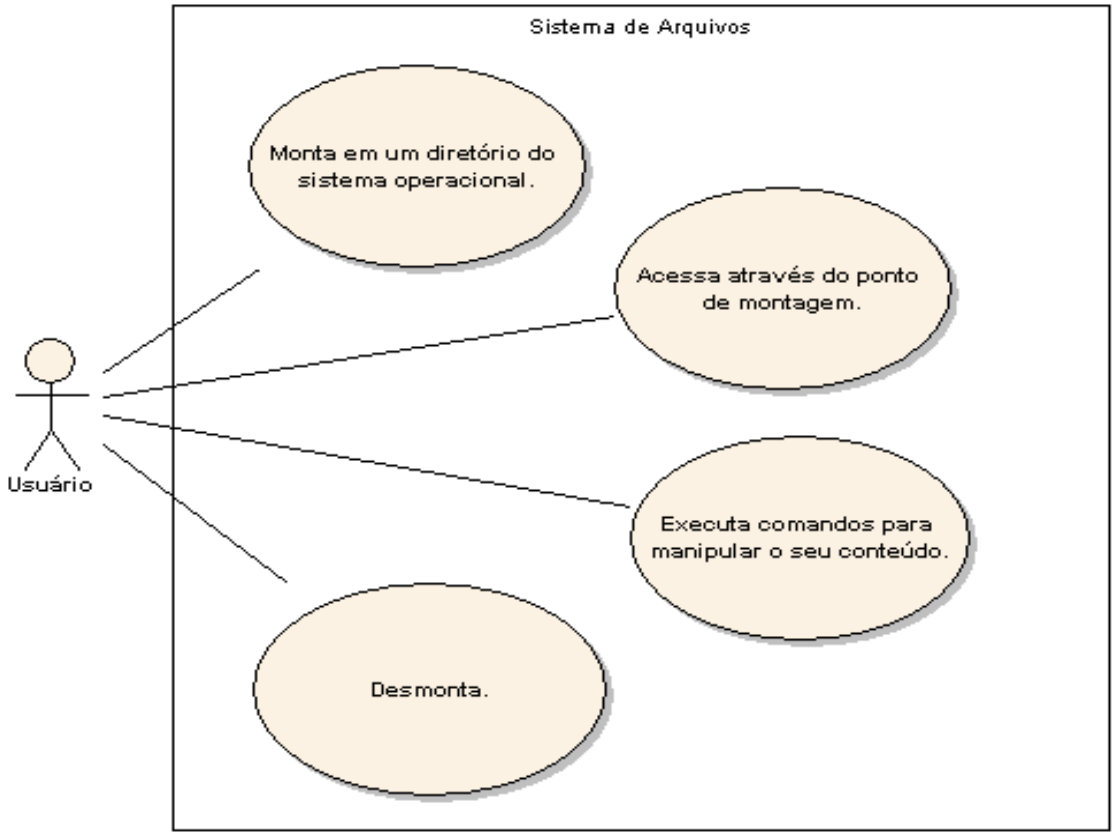
- **Funcionais:**
  - permitir a montagem do sistema de arquivos em um diretório do sistema operacional;
  - permitir que os arquivos e diretórios que forem criados ou copiados para o ponto de montagem, sejam armazenados dentro de um arquivo (encapsulamento);
  - permitir que o encapsulamento gerado possa ser montado em um diretório do sistema operacional, para manipulação de seu conteúdo.

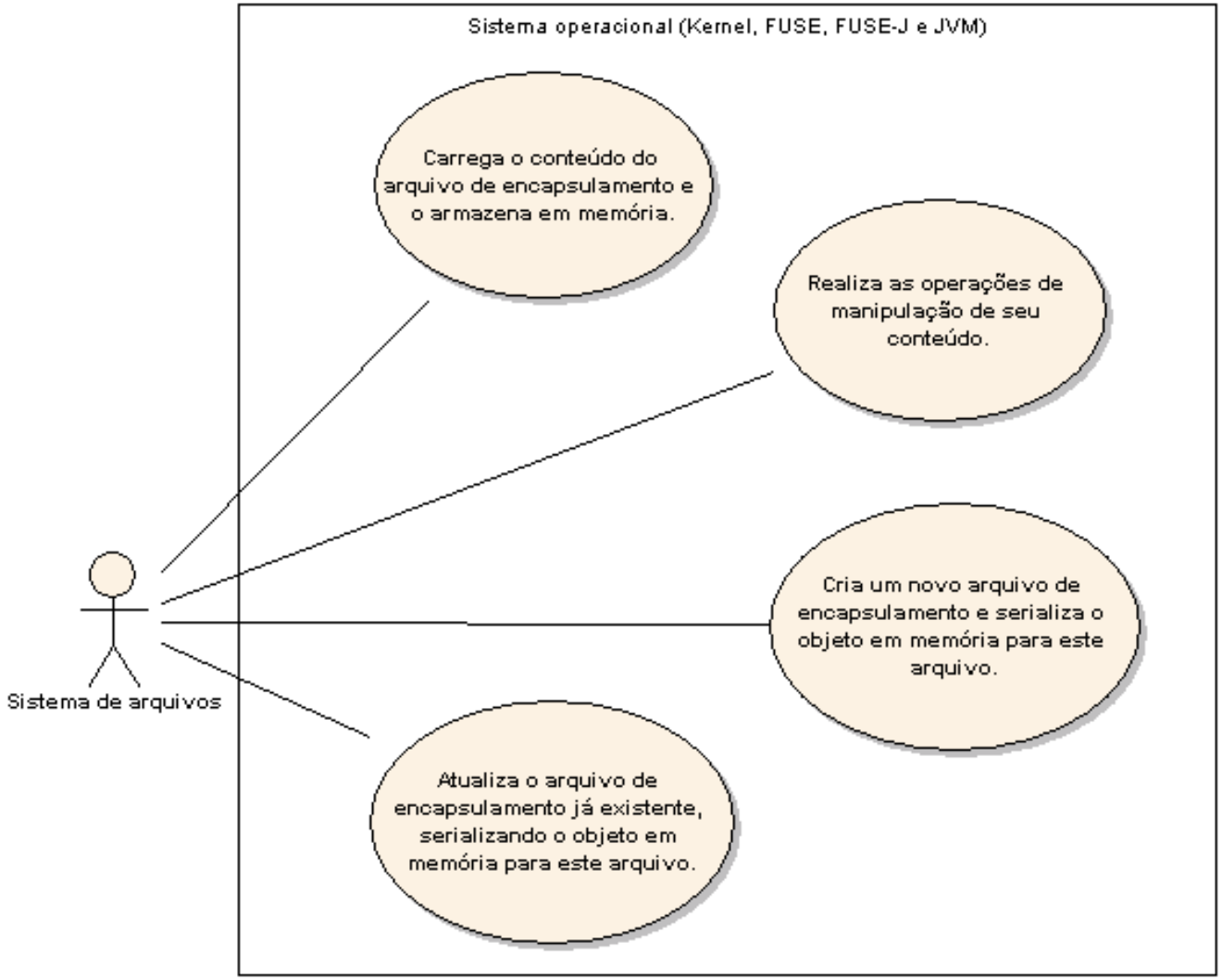
- **Não-funcionais:**
  - ser implementado sobre a plataforma FUSE;
  - ser implementado utilizando o ambiente NetBeans IDE 6.7;
  - ser implementado na linguagem Java, versão 1.5;
  - ser implementado utilizando a interface FUSE-J;
  - ser compatível com as distribuições GNU/Linux.

3.2. Especificação

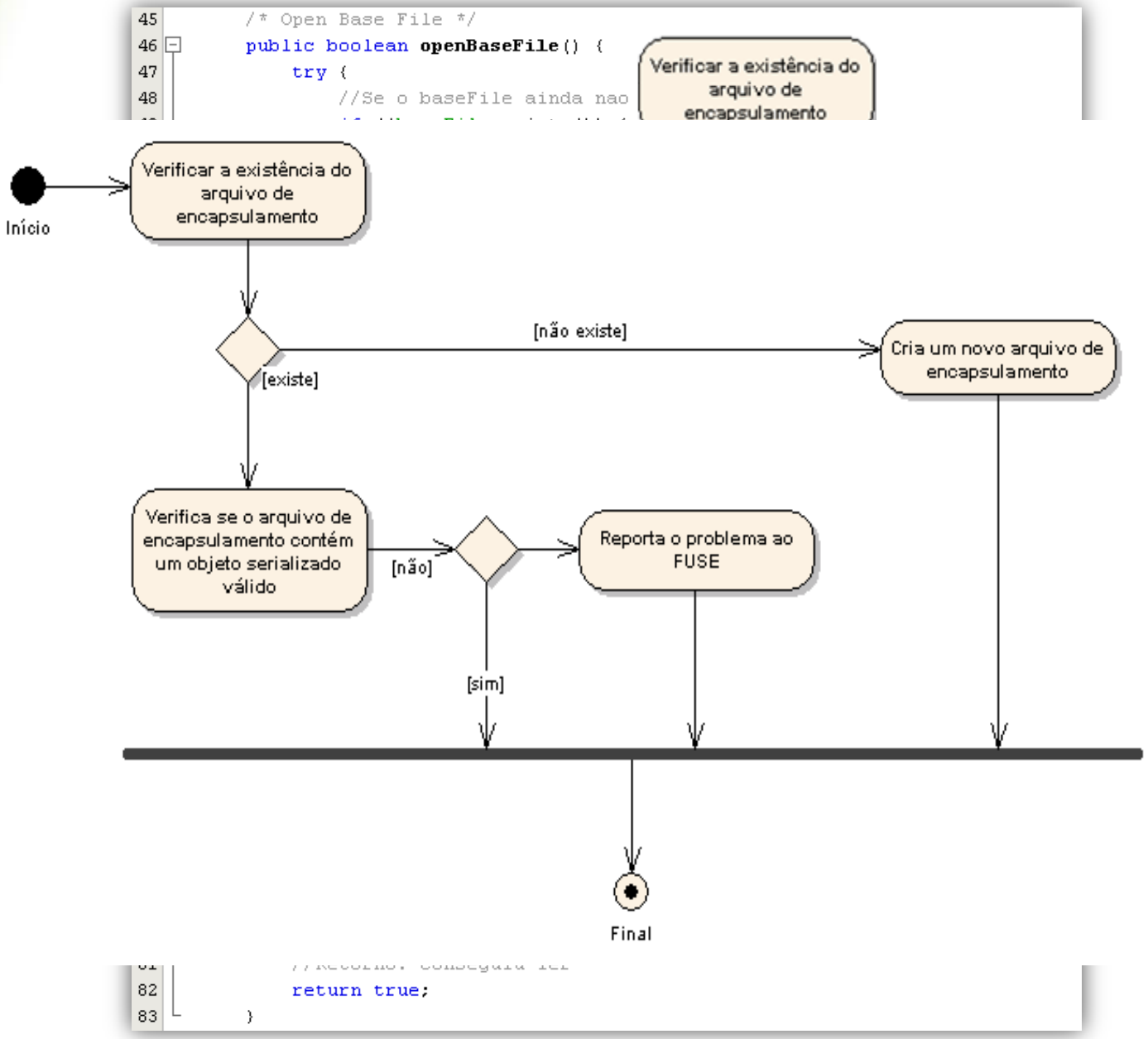
– Ferramenta: Enterprise Architect 5.0 (Sparx Systems).

3.2.1. DIAGRAMAS DE CASOS DE USO:

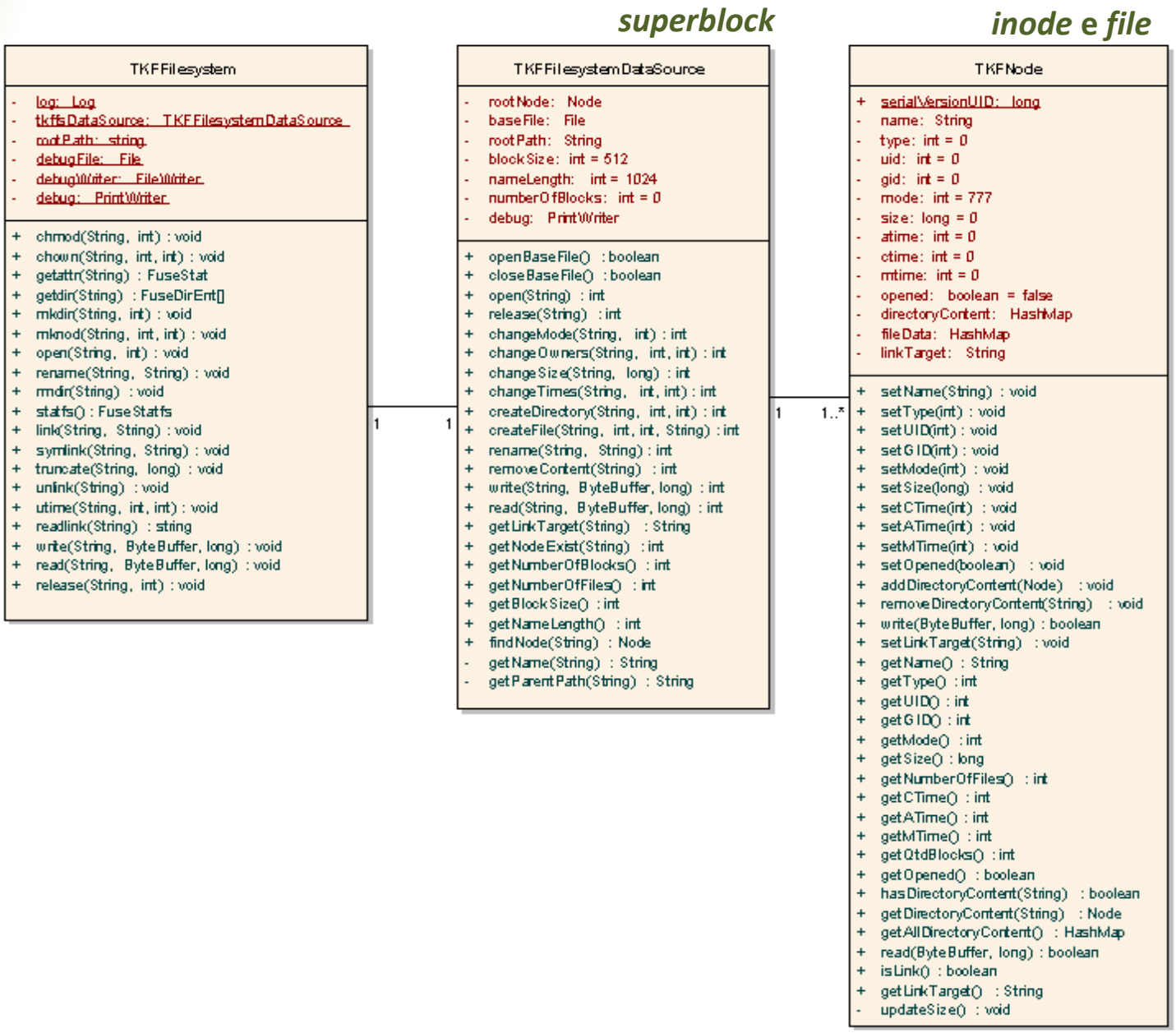



Sistema operacional (Kernel, FUSE, FUSE-J e JVM)
Carrega o conteúdo do arquivo de encapsulamento e o armazena em memória.
Realiza as operações de manipulação de seu conteúdo.
Cria um novo arquivo de encapsulamento e serializa o objeto em memória para este arquivo.
Atualiza o arquivo de encapsulamento já existente, serializando o objeto em memória para este arquivo.
Sistema de arquivos

3.2.2. DIAGRAMA DE ATIVIDADES:



3.2.3. DIAGRAMA DE CLASSES:

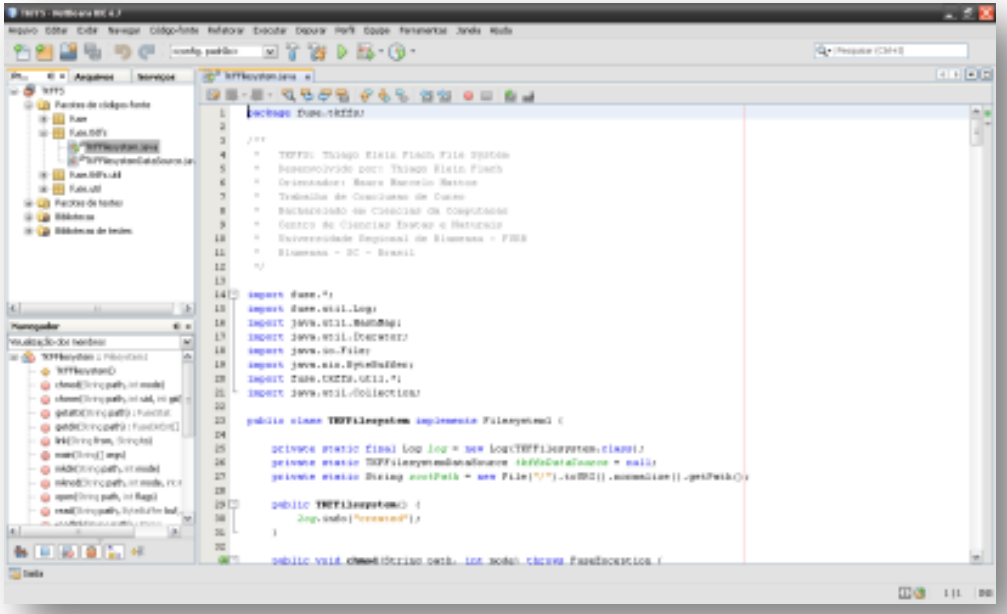


3.3. Implementação

3.3.1. TÉCNICAS E FERRAMENTAS UTILIZADAS:

Ferramentas:

- NetBeans IDE 6.7;
- Java JDK 1.5;
- FUSE-J.



3.3.1.1. Classe TKFFilesystem

- Implementa a interface do FUSE-J:

```
import java.util.Collection;

public class TKFFilesystem implements Filesystem1 {

    private static final Log log = new Log(TKFFilesystem.class);
    private static TKFFilesystemDataSource tkffsDataSource = null;
```

- Inicializa o sistema de arquivos (carrega o arquivo de encapsulamento e o mapeia no diretório) ou o finaliza (atualiza o arquivo de encapsulamento):

```
File tkfFile = new File(pathTKFFile);

tkffsDataSource = new TKFFilesystemDataSource(rootPath, tkfFile);

//Se conseguir ler o baseFile
if (tkffsDataSource.openBaseFile()) {
    log.info("entering");

    try {
        FuseMount.mount(fuseArgs, new TKFFilesystem());
    } catch (Exception e) {
        log.error(e.getMessage());
    } finally {
        //Se nao conseguir fechar o basFile
        if (tkffsDataSource.closeBaseFile()) {
            log.info("exiting");
        } else {
            log.error("impossible to write the base file");
        }
    }
} else {
    log.error("impossible to read the base file");
}
```

- Método write:

```
405     public void write(String path, ByteBuffer buf, long offset) throws FuseException {
406         //Normaliza os paths
407         String normPath = new File(path).toURI().normalize().getPath();
408 
409         //Tenta escrever
410         switch (tkffsDataSource.write(normPath, buf, offset)) {
411             /* case 0: {} --- OK */
412             case 1: {
413                 throw new FuseException("No such device or address").initErrno(FuseException.ENXIO);
414             }
415             case 2: {
416                 throw new FuseException("No such file or directory").initErrno(FuseException.ENOENT);
417             }
418             case 3: {
419                 throw new FuseException("Is a directory").initErrno(FuseException.EISDIR);
420             }
421             case 4: {
422                 throw new FuseException("I/O error").initErrno(FuseException.EIO);
423             }
424         }
425     }
```

- Método read:

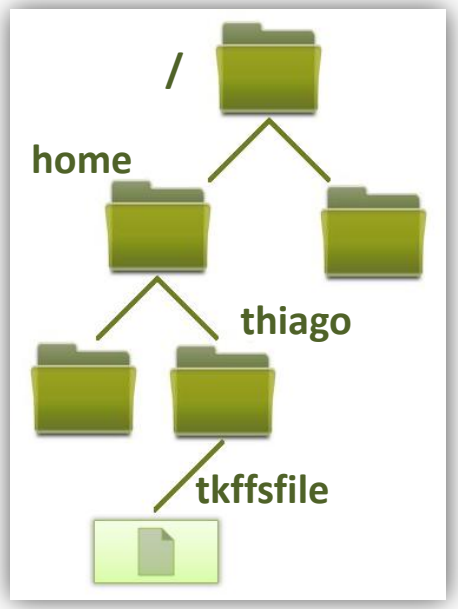
```
427     public void read(String path, ByteBuffer buf, long offset) throws FuseException {
428         //Normaliza os paths
429         String normPath = new File(path).toURI().normalize().getPath();
430 
431         //Tenta ler
432         switch (tkffsDataSource.read(normPath, buf, offset)) {
433             /* case 0: {} --- OK */
434             case 1: {
435                 throw new FuseException("No such device or address").initErrno(FuseException.ENXIO);
436             }
437             case 2: {
438                 throw new FuseException("No such file or directory").initErrno(FuseException.ENOENT);
439             }
440             case 3: {
441                 throw new FuseException("Is a directory").initErrno(FuseException.EISDIR);
442             }
443             case 4: {
444                 throw new FuseException("I/O error").initErrno(FuseException.EIO);
445             }
446         }
447     }
```

3.3.1.2. Classe TKFFilesystemDataSource

- Gerencia os dados armazenados no arquivo de encapsulamento, fazendo buscas na árvore de diretórios, a fim de realizar as alterações no nó correto;

- A busca se dá de acordo com o caminho passado como parâmetro através do FUSE.

Exemplo: **/home/thiago/tkffsfile**



- Método findNode():

```
598         /*
599          * Este metodo procura o arquivo/diretório em questão
600          */
601         public TKFNode findNode(String path) {
602             //Pega o node raiz
603             TKFNode nodeFinded = rootNode;
604 
605             //Armazena o path sem o / inicial
606             String pathWithoutRoot = path.substring(1, path.length());
607 
608             //Armazena todos os nodes do path em um array
609             String[] nodeNames = pathWithoutRoot.split("\\/");
610 
611             //Para todos os nodes do path
612             for (int i = 0; i < nodeNames.length; i++) {
613 
614                 //Se nodeAux tiver o filho a ser procurado
615                 if (nodeFinded.hasDirectoryContent(nodeNames[i])) {
616 
617                     nodeFinded = nodeFinded.getDirectoryContent(nodeNames[i]);
618                 } else {
619                     //Sai do loop
620                     break;
621                 }
622             }
623 
624             //Se o utltimo node encontrado for o procurado
625             if (nodeFinded.getName().equals(getName(path))) {
626                 return nodeFinded;
627             } else {
628 
629                 //Retorno: arquivo ou diretorio nao existe
630                 return null;
631             }
632         }
```

- Método openBaseFile():

```
45      /* Open Base File */
46      public boolean openBaseFile() {
47          try {
48              //Se o baseFile ainda nao existir
49              if (!baseFile.exists()) {
50                  //Cria
51                  closeBaseFile();
52              }
53
54              //Carrega o aruivo usando FileInputStream
55              FileInputStream f_in = new FileInputStream(baseFile);
56
57              //Carrega o objeto usando ObjectInputStream
58              ObjectInputStream obj_in = new ObjectInputStream(f_in);
59
60              //Carrega o objeto
61              Object obj = obj_in.readObject();
62
63              //Se o objeto for um TKFNode
64              if (obj instanceof TKFNode) {
65                  //Faz um Cast do objeto para TKFNode
66                  rootNode = (TKFNode) obj;
67              } else {
68                  //Retorno: nao e um objeto valido
69                  return false;
70              }
71
72              obj_in.close();
73              f_in.close();
74          } catch (Exception e) {
75              e.printStackTrace();
76
77              //Retorno: nao conseguiu ler
78              return false;
79          }
80
81          //Retorno: conseguiu ler
82          return true;
83      }
```

- Método closeBaseFile():

```
85      /* Close Base File */
86      public boolean closeBaseFile() {
87          try {
88              //Escreve o arquivo com FileOutputStream
89              FileOutputStream f_out = new FileOutputStream(baseFile);
90
91              //Escreve o objeto com ObjectOutputStream
92              ObjectOutputStream obj_out = new ObjectOutputStream(f_out);
93
94              //Escreve o objeto no arquivo
95              obj_out.writeObject(rootNode);
96
97              obj_out.flush();
98              obj_out.close();
99              f_out.close();
100         } catch (Exception e) {
101             e.printStackTrace();
102
103             //Retorno: nao conseguiu escrever
104             return false;
105         }
106
107         //Retorno: conseguiu escrever
108         return true;
109     }
```

- Método write():

```
443         /* write */
444         public int write(String path, ByteBuffer buf, long offset) {
445             //Se o path comecar com /
446             if (path.charAt(0) == '/') {
447 
448                 TKFNode nodeFinded = findNode(path);
449 
450                 //Se encontrou o node
451                 if (nodeFinded != null) {
452 
453                     //Se o node for um arquivo
454                     if (nodeFinded.getType() == 0) {
455 
456                         //Se nao conseguir escrever o conteudo
457                         if (!nodeFinded.write(buf, offset)) {
458                             return 4;
459                         }
460                     } else{
461                         //Se o node for um link
462                         if (nodeFinded.getType() == 2) {
463 
464                             //Tenta escrever o conteudo
465                             return write(nodeFinded.getLinkTarget(), buf, offset);
466                         } else {
467                             //Retorno: e um diretorio
468                             return 3;
469                         }
470                     }
471                 } else {
472                     //Retorno: arquivo ou diretorio nao existe
473                     return 2;
474                 }
475             } else {
476                 //Retorno: nao e um caminho valido
477                 return 1;
478             }
479 
480             return 0;
481         }
```

- Método read():

```
483         /* read */
484         public int read(String path, ByteBuffer buf, long offset) {
485             //Se o path comecar com /
486             if (path.charAt(0) == '/') {
487 
488                 TKFNode nodeFinded = findNode(path);
489 
490                 //Se encontrou o node
491                 if (nodeFinded != null) {
492 
493                     //Se o node for um arquivo
494                     if (nodeFinded.getType() == 0) {
495 
496                         //Se nao conseguir ler o conteudo
497                         if (!nodeFinded.read(buf, offset)) {
498                             //Retorno: erro de I/O
499                             return 4;
500                         }
501                     }else{
502                         //Se o node for um link
503                         if (nodeFinded.getType() == 2) {
504                             //Tenta ler o conteudo
505                             return read(nodeFinded.getLinkTarget(), buf, offset);
506                         } else {
507                             //Retorno: e um diretorio
508                             return 3;
509                         }
510                     }
511                 } else {
512                     //Retorno: arquivo ou diretorio nao existe
513                     return 2;
514                 }
515             } else {
516                 //Retorno: nao e um caminho valido
517                 return 1;
518             }
519 
520             return 0;
521         }
```

3.3.1.3. Classe TKFNode

- Representa os nós da árvore de diretórios (arquivos, diretórios e links);
- Armazena o conteúdo de um arquivo, de um diretório ou o alvo de um link;
- Não separa dados de metadados (diferente dos inodes);
- Seus objetos são serializados para serem gravados no arquivo de encapsulamento:

```
public class TKFNode implements Serializable {
```

- Método write():

```
253     public boolean write(ByteBuffer dataBuffer, long offset) {
254         try {
255             Long lg = new Long(offset);
256 
257             byte[] bytes = new byte[dataBuffer.capacity()];
258             dataBuffer.get(bytes, 0, bytes.length);
259 
260             fileData.put(lg, bytes);
261 
262             //Atualiza o tamanho do arquivo
263             updateSize();
264 
265         } catch (Exception e) {
266             e.printStackTrace();
267 
268             return false;
269         }
270 
271         return true;
272     }
```

- Método read():

```
232         /*
233          * METODOS ESPECIAIS
234          */
235
236         public boolean read(ByteBuffer dataBuffer, long offset) {
237
238             try {
239                 Long lg = new Long(offset);
240
241                 byte[] bytes = (byte[])fileData.get(lg);
242
243                 dataBuffer.put(bytes, 0, bytes.length);
244             } catch (Exception e) {
245                 e.printStackTrace();
246
247                 return false;
248             }
249
250             return true;
251         }
```

3.3.2. OPERACIONALIDADE DA IMPLEMENTAÇÃO:

- Procedimento de instalação:

1. Conectar no sistema com o usuário **root**;
2. **Descompactar** o pacote de instalação do TKFFS;
3. **Entrar no diretório** gerado pela descompactação;
4. Editar o arquivo de configuração **build.conf**:

```
#!/bin/bash
# by: Thiago Klein Flach

# JDK 1.4 or greater HOME
export JDK_HOME="/home/thiago/jdk1.5.0_20"

# FUSE library & headers base directory
export FUSE_HOME="/usr/local"
```

5. Executar o script **install.sh**:

```
root@thiagotcc:/usr/src/tkffs-2009# ./install.sh
CONFIG FILE /etc/tkffs.conf CREATED
LOG FILE /var/log/tkffs.log CREATED
"/home/thiago/jdk1.5.0_20"/bin/javac -Xlint -source 1.4 -d build src/fuse/Filesy
stemToFuseFSAdapter.java src/fuse/Filesystem1ToFilesystemAdapter.java src/fuse/F
useStat.java src/fuse/FuseException.java src/fuse/FuseStatfs.java src/fuse/util/
Struct.java src/fuse/util/Log.java src/fuse/Filesystem.java src/fuse/FuseDirEnt.
java src/fuse/FuseMount.java src/fuse/FuseFS.java src/fuse/Filesystem1.java src/
fuse/FuseFSDirEnt.java src/fuse/FilesystemConstants.java src/fuse/FakeFilesystem
.java src/fuse/tkffs/util/TKFNode.java src/fuse/tkffs/TKFFilesystemDataSource.ja
va src/fuse/tkffs/TKFFilesystem.java src/fuse/FuseFtype.java src/java2c/CAPIGene
rator.java src/java2c/DumpJVMLdPath.java
make -C jni all
make[1]: Entrando no diretÃ³rio `/usr/src/tkffs-2009/jni'
make[1]: Nada a ser feito para `all'.
make[1]: Saindo do diretÃ³rio `/usr/src/tkffs-2009/jni'

TKFFS IS COMPILED AND BUILT!

root@thiagotcc:/usr/src/tkffs-2009#
```

6. Editar o arquivo de configuração **/etc/tkffs.conf**:

```
#!/bin/bash
# TKFFS config file
# by: Thiago Klein Flach

# TKFFS home
export TKFFS_HOME="/usr/src/tkffs-2009"

# TKFFS log file
export TKFFS_LOG="/var/log/tkffs.log"
```

- Script install.sh:

```
 1 #!/bin/bash
 2 # by: Thiago Klein Flach
 3
 4 TKFFS_CONF="/etc/tkffs.conf"
 5 TKFFS_LOG="/var/log/tkffs.log"
 6
 7 #Cria o arquivo de configuracao
 8 echo "#!/bin/bash" > $TKFFS_CONF
 9 echo "# TKFFS config file" >> $TKFFS_CONF
10 echo "# by: Thiago Klein Flach" >> $TKFFS_CONF
11 echo "" >> $TKFFS_CONF
12 echo "# TKFFS home" >> $TKFFS_CONF
13 echo "export TKFFS_HOME=\"\"" >> $TKFFS_CONF
14 echo "" >> $TKFFS_CONF
15 echo "# TKFFS log file" >> $TKFFS_CONF
16 echo "export TKFFS_LOG=\"$TKFFS_LOG\"" >> $TKFFS_CONF
17
18 #Define permissao total de acesso ao arquivo
19 chmod 777 $TKFFS_CONF
20
21 #OUTPUT
22 echo "CONFIG FILE $TKFFS_CONF CREATED"
23
24 #Tenta criar o arquivo log e definir suas permissoes
25 echo "" > $TKFFS_LOG && chmod 766 $TKFFS_LOG
26
27 #OUTPUT
28 echo "LOG FILE $TKFFS_LOG CREATED"
29
30 #Tenta compilar o codigo
31 make
```

- Script tkffsmount:

```
#!/bin/bash
# by: Thiago Klein Flach

TKFFS_CONF="/etc/tkffs.conf"

# Se o arquivo de configuracao existe
if [ -e $TKFFS_CONF ]; then

        # Tenta executar o arquivo de configuracao
        . $TKFFS_CONF

        # Se conseguiu executar o arquivo de configuracao
        if [ 0 -eq $? ]; then
                # Se recebeu 2 parametros
                if [ 2 -eq $# ]; then
                        # Se o primeiro parametro for um diretorio
                        if [ -d $1 ]; then
                                # Remove a / do final
                                DIR=`echo $1 | sed s/"\/$"/""/g`

                                # Executa o comando para verificar se a montagem foi realizada
                                IS_MOUNTED=`cat /etc/mtab | grep $DIR | wc -l`

                                # Se NAO  houver algo ja mapeado no diretorio
                                if [ 0 -eq $IS_MOUNTED ]; then
                                        # Se o segundo parametro for um arquivo
                                        nohup sh $TKFFS_HOME/tkffsengine $1 $2 2>> $TKFFS_LOG &

                                        # Aguarda montar
                                        sleep 4

                                        # Executa o comando para verificar se a montagem foi realizada
                                        IS_MOUNTED=`cat /etc/mtab | grep $DIR | grep "fuse.javafs" | wc -l`

                                        # Se esta montado
                                        if [ $IS_MOUNTED -gt 0 ]; then
                                                echo "TKFFS IS MOUNTED!"
                                        else
                                                echo "TKFFS can NOT be mounted"
                                                echo "SEE: $TKFFS_LOG"
                                                tail $TKFFS_LOG
                                        fi
                                else
                                        echo "TKFFS can NOT be mounted"
                                        echo "the directory $DIR is busy"
                                fi
                        else
                                echo "$1: is NOT a directory"
                        fi
                else
                        echo "USAGE: tkffsmount [mount_point] [base_file]"
                fi
        else
                echo "the config file $TKFFS_CONF can NOT be executed"
        fi
else
        echo "the config file $TKFFS_CONF NOT exist"
fi
```

- Script tkffsumount:

```
#!/bin/bash
# by: Thiago Klein Flach

TKFFS_CONF="/etc/tkffs.conf"

# Se o arquivo de configuracao existe
if [ -e $TKFFS_CONF ]; then

        # Tenta executar o arquivo de configuracao
        . $TKFFS_CONF

        # Se conseguiu executar o arquivo de configuracao
        if [ 0 -eq $? ]; then
                # Se recebeu um parametro
                if [ 1 -eq $# ]; then
                        # Se o parametro for para mostrar o HELP
                        if [ $1 = '-h' ]; then
                                echo "usage: tkffsumount [mount_point]"
                        else
                                # Se o parametro for um diretorio
                                if [ -d $1 ]; then
                                        # Remove a / do final
                                        DIR=`echo $1 | sed s/"\/$"/""/g`

                                        # Executa o comando para verificar se o diretorio esta montado
                                        IS_MOUNTED=`cat /etc/mtab | grep $DIR | grep "fuse.javafs" | wc -l`

                                        # Se esta montado
                                        if [ $IS_MOUNTED -gt 0 ]; then
                                                # Tenta desmontar
                                                fusermount -u $DIR >> $TKFFS_LOG

                                                # Se foi desmontado com sucesso
                                                if [ 0 -eq $? ]; then
                                                        echo "TKFFS IS UMOUNTED!"
                                                else
                                                         echo "TKFFS can NOT be umounted"
                                                         echo "SEE: $TKFFS_LOG"
                                                         tail $TKFFS_LOG
                                                fi
                                        else
                                                echo "TKFFS is NOT mounted in $DIR"
                                        fi
                                else
                                        echo "$1: is NOT a directory"
                                fi
                        fi
                else
                        echo "USAGE: tkffsumount [mount_point]"
                fi
        else
                echo "the config file $TKFFS_CONF can NOT be executed"
        fi
else
        echo "the config file $TKFFS_CONF NOT exist"
fi
```

- Script tkffsengine:

```
#!/bin/sh
# by: Thiago Klein Flach

# Usage: ./tkffs_mount.sh /mount/point file

TKFFS_CONF="/etc/tkffs.conf"

# Se o arquivo de configuracao existe
if [ -e $TKFFS_CONF ]; then

        # Tenta executar o arquivo de configuracao
        . $TKFFS_CONF

        # Se conseguiu executar o arquivo de configuracao
        if [ 0 -eq $? ]; then

                # Tenta executar o arquivo build.conf
                . $TKFFS_HOME/build.conf

                # Se conseguiu executar
                if [ 0 -eq $? ]; then

                        LD_LIBRARY_PATH=$TKFFS_HOME/jni:$FUSE_HOME/lib $JDK_HOME/bin/java -classpath $TKFFS_HOME/build fuse.tkffs.TKFFilesystem -f -s $1 $2
                else
                        echo "the file $TKFFS_HOME/build.conf can NOT be executed"
                fi
        else
                echo "the config file $TKFFS_CONF can NOT be executed"
        fi
else
        echo " the config file $TKFFS_CONF NOT exist"
fi
```

- Utilização do TKFFS

CASO 1: O usuário monta um novo arquivo de encapsulamento em um diretório **vazio** do sistema operacional e realiza operações para manipulação:

```
thiago@thiagotcc:~$ tkffsmount /home/thiago/mnt tkffsfile
TKFFS IS MOUNTED!
thiago@thiagotcc:~$ cd /home/thiago/mnt/
thiago@thiagotcc:~/mnt$ ls -lah
total 0
thiago@thiagotcc:~/mnt$ mkdir diret1
thiago@thiagotcc:~/mnt$ mkdir diret2
thiago@thiagotcc:~/mnt$ ls -lah
total 0
drwxr-xr-x 0 root root 0 2009-11-20 15:31 diret1
drwxr-xr-x 0 root root 0 2009-11-20 15:31 diret2
thiago@thiagotcc:~/mnt$ cd diret1
thiago@thiagotcc:~/mnt/diret1$ touch arq
thiago@thiagotcc:~/mnt/diret1$ ls -lah
total 0
-rw-r--r-- 0 root root 0 2009-11-20 15:32 arq
thiago@thiagotcc:~/mnt/diret1$ chmod 664 arq
thiago@thiagotcc:~/mnt/diret1$ ls -lah
total 0
-rw-rw-r-- 0 root root 0 2009-11-20 15:32 arq
thiago@thiagotcc:~/mnt/diret1$ vi arq
thiago@thiagotcc:~/mnt/diret1$ cat arq
Linha 1
Linha 2
Linha 3
thiago@thiagotcc:~/mnt/diret1$ cd ../diret2
thiago@thiagotcc:~/mnt/diret2$ ln ../diret1/arq lin
thiago@thiagotcc:~/mnt/diret2$ ls -lah
total 512
-rw-rw-r-- 0 root root 24 2009-11-20 15:32 lin
thiago@thiagotcc:~/mnt/diret2$ cat lin
Linha 1
Linha 2
Linha 3
thiago@thiagotcc:~/mnt/diret2$ tail -1 lin
Linha 3
thiago@thiagotcc:~/mnt/diret2$ cd ../..
thiago@thiagotcc:~$ tkffsumount /home/thiago/mnt
TKFFS IS UMOUNTED!
thiago@thiagotcc:~$
```

CASO 2: O usuário monta o arquivo de encapsulamento criado no caso anterior, no mesmo ponto de montagem e realiza outras operações para manipulação :

```
thiago@thiagotcc:~$ tkffsmount /home/thiago/mnt tkffsfile
TKFFS IS MOUNTED!
thiago@thiagotcc:~$ tar -xzvf tkffs-2009.tgz -C /home/thiago/mnt
tkffs-2009/
tkffs-2009/src/
tkffs-2009/src/fuse/
tkffs-2009/src/fuse/FilesystemToFuseFSAdapter.java
tkffs-2009/src/fuse/Filesystem1ToFilesystemAdapter.java
tkffs-2009/jni/javafs bindings.o
(...)
tkffs-2009/build.conf
tkffs-2009/tkffsengine
tkffs-2009/tkffsumount
tkffs-2009/tkffsmount
tkffs-2009/nohup.out
tkffs-2009/fuse.iws
thiago@thiagotcc:~$ cd /home/thiago/mnt
thiago@thiagotcc:~/mnt$ ls -lah
total 0
drwxr-xr-x 0 root root   24 2009-11-20 15:31 diret1
drwxr-xr-x 0 root root    0 2009-11-20 15:31 diret2
drwxr-xr-x 0 root root 446K 2009-11-20 08:00 tkffs-2009
thiago@thiagotcc:~/mnt$ du -sh *
512     diret1
512     diret2
97K     tkffs-2009
thiago@thiagotcc:~/mnt$ rm -rf diret2
thiago@thiagotcc:~/mnt$ ls -lah
total 0
drwxr-xr-x 0 root root   24 2009-11-20 15:31 diret1
drwxr-xr-x 0 root root 446K 2009-11-20 08:00 tkffs-2009
thiago@thiagotcc:~/mnt$ cd ..
thiago@thiagotcc:~$ tkffsmount /home/thiago/mnt tkffsfile
TKFFS can NOT be mounted
the directory /home/thiago/mnt is busy
thiago@thiagotcc:~$ tkffsumount /home/thiago/mnt
TKFFS IS UMOUNTED!
thiago@thiagotcc:~$ ls -lah tkffsfile
-rw-r--r-- 1 thiago thiago 459K 2009-11-20 15:52 tkffsfile
thiago@thiagotcc:~$
```

3.4. Resultados e discussão

Sistema	Mapeamento	Rede	Dispositivo			Serviço	FUSE	FUSE-J
			Físico	Virtual	Móvel			
MOOFS	X	X	X				X	
BloggerFS	X	X				X	X	X
ClamFS	X		X			X	X	
TKFFS	**X**			**X**	**X**		**X**	**X**

Analogia:

Arquivo de encapsulamento (móvel e virtual)

Pen drive (móvel)

4 - Conclusão

- **Dificuldades iniciais;**
  (APIs complexas)

- **Por que serialização?**
  (Tempo e facilidade de abstração).

- **Por que gravar para o arquivo somente no final?**
  (I/O em memória)

- **O primeiro passo foi dado!**

4.1. Extensões

- **Armazenamento em blocos;**
- **Compactação;**
- **Criptografia;**
- **Migração dos dados entre máquinas, sobre TCP/IP.**