Creating a File System Manager using Haskell

Cristian Castillo McQuiddy **(2014061245)**
Jairo Mendez Martnez **(2014050475)**

Instituto Tecnolgico de Costa Rica

As part of this project a file system is going to be written in haskell to emulate functionality a file manager covering volumes, file systems and files management processes in Linux.

0 INTRODUCTION

On a UNIX system, everything is a file; if something is not a file, it is a process.A Linux system, just like UNIX, makes no difference between a file and a directory, since a directory is just a file containing names of other files. Programs, services, texts, images, and so forth, are all files. Input and output devices, and generally all devices, are considered to be files, according to the system, [1].

The Filesystem Hierarchy Standard (FHS) defines the main directories and their contents in Linux operating systems. For the most part, it is a formalization and extension of the traditional BSD filesystem hierarchy, as illustrated in Figure 1. [2].
.

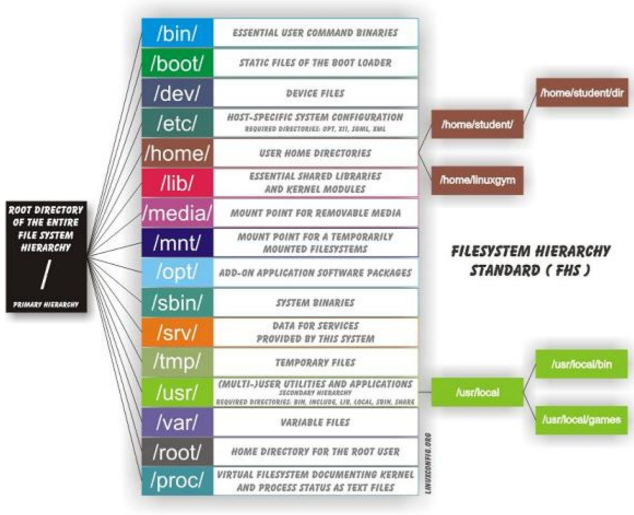

ROOT DIRECTORY OF THE ENTIRE FILE SYSTEM HIERARCHY
/
PRIMARY HIERARCHY
/bin/
ESSENTIAL USER COMMAND BINARIES
/boot/
STATIC FILES OF THE BOOT LOADER
/dev/
DEVICE FILES
/etc/
HOST-SPECIFIC SYSTEM CONFIGURATION
REQUIRED DIRECTORIES: OPT, X11, SGML, XML
/home/
USER HOME DIRECTORIES
/lib/
ESSENTIAL SHARED LIBRARIES AND KERNEL MODULES
/media/
MOUNT POINT FOR REMOVABLE MEDIA
/mnt/
MOUNT POINT FOR A TEMPORARILY MOUNTED FILESYSTEMS
/opt/
ADD-ON APPLICATION SOFTWARE PACKAGES
/sbin/
SYSTEM BINARIES
/srv/
DATA FOR SERVICES PROVIDED BY THIS SYSTEM
/tmp/
TEMPORARY FILES
/usr/
(MULTI-)USER UTILITIES AND APPLICATIONS
SECONDARY HIERARCHY
REQUIRED DIRECTORIES: BIN, INCLUDE, LIB, LOCAL, SBIN, SHARE
/var/
VARIABLE FILES
/root/
HOME DIRECTORY FOR THE ROOT USER
/proc/
VIRTUAL FILESYSTEM DOCUMENTING KERNEL AND PROCESS STATUS AS TEXT FILES
/home/student/
/home/student/dir
/home/linuxgym
FILESYSTEM HIERARCHY STANDARD (FHS)
/usr/local
/usr/local/bin
/usr/local/games
LINUXCONFIG.ORG

Fig. 1. File System Hierarchy Standard (FHS).

Linux where made as multi-user systems t unlike Windows was created for a single user.

A file system is used to control how data is stored and retrieved. Without a file system, information placed in a storage area would be one large body of data with no way to tell where one piece of information stops and the next begins. By separating the data into pieces and giving each piece a name, the information is easily isolated and identified. Taking its name from the way paper-based information systems are named, each group of data is called a ”file”. The structure and logic rules used to manage the groups of information and their names is called a **”file system”**.[2]

Files have ownership composed by users and groups. Hence as part of this system, we will be able to manage basic **user and groups** to be able to setup the ownership in the file and directories that will be created in this system.[3]

Linux uses more than one partition on the same disk.One of the goals of having different partitions is to achieve higher data security in case of disaster. By dividing the hard disk in partitions, data can be grouped and separated. When an accident occurs, only the data in the partition that got the hit will be damaged, while the data on the other partitions will most likely survive.[1]

Storage devices will be created to emulate the equivalent to SAN LUNs, hard drives or any other storage device required to create a file system or a logical volume.Each storage device could initialize to be used by **LVM**[2]

As illustrated in Figure 2, logical volumes manager allow us to create a set of volumes where the file systems are going to be created over. The volume groups are formed by physical volumes (or storage devices) and volumes. A volume can be created using the total space allocated in the volume group and not to be limited by the size of a single storage device.[2]

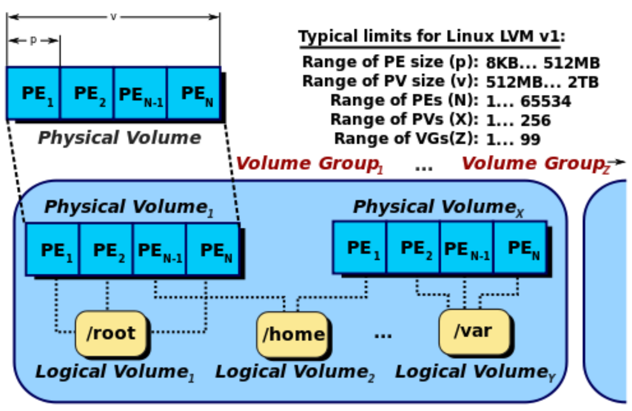

Typical limits for Linux LVM v1:
Range of PE size (p): 8KB... 512MB
Range of PV size (v): 512MB... 2TB
Range of PEs (N): 1... 65534
Range of PVs (X): 1... 256
Range of VGs(Z): 1... 99
Physical Volume
Volume Group1 ... Volume GroupZ
Physical Volume1
Physical VolumeX
PE1 PE2 PEN-1 PEN
/root
/home
/var
Logical Volume1 Logical Volume2 Logical VolumeY

Fig. 2. LVM (Logical Volume Manager).[4]

The file systems are going to be created over a single logical volume or over a specific storage device.

The file system mount points are directories associated with an existent file system. Both the file systems and mount points need to be created in order to be able to associate them with each other. To be able to use a file system to create files or directories inside of it, it needs to be mounted and mapped over an existent directory.[2]

Haskell is a standardized, general-purpose purely functional programming language, with non-strict semantics and strong static typing. It was designed without any application niche in mind. Although it takes a strong stand on how programs should be written, it does not favor one problem domain over others. While at its core, the language encourages a pure, lazy style of functional programming, this is the default, not the only option.

.

The program is going to interact with the user over the command line. The user will be triggering commands to be able to interact with the program. The program is go to display the outputs of each of the operations requested by a user . The commands are going to impact the environment that is going to be managed by the file manager.[2]

1 Development environment

The file system was implemented in Linux , and was developed using haskell language, which is an functional language, this one can be implemented in any haskell IDLE, for the development for this investigation SublimeText3 is the one that was used, this one can be downloaded in [5].

2 Libraries

- import Data.Tuple.Select : Used for the structures of the file system
- import Data.List.Split : Used to split some symbols like spaces " ".
- import Control.Monad : Library that control the secuense of the program, like the command forever.

3 Data Structures

a) tupleGetFirst' : Get the first element of the group tuple.

$\&(x,\_,\_)$:Only matters the first element.

b) tupleGetSecond' : Get the second element of the group tuple.

$\&(\_,y,\_)$:Only matters the second element.

c) tupleGetThird' : Get the third element of the group tuple.

$\&(\_,\_,z)$:Only matters the third element.

d) tupleGetFirst : Get the first element of the user tuple.

$\&(x,\_,\_,\_)$:Only matters the first element.

e) tupleGetSecond : Get the second element of the user tuple.

$\&(\_,y,\_,\_)$:Only matters the second element.

f) tupleGetThird : Get the third element of the user tuple.

$\&(\_,\_,z,\_)$:Only matters the third element.

g) tupleGetFourth : Get the fourth element of the user tuple.

$\&(\_,\_,\_,a)$:Only matters the fourth element .

4 Functions

a) fsManager () : This function take the instructions from the FSManagerConsole (interfaz) and starts calling function that will modify the list according to the input instruction .This functions calls itself recursively with the Total parameteres modified.

This function have the instruction $<-$ **getLine** , where is the only way to get the text or the commands that the user input to the program.

Other function that have is $<-$ **getZonedTime**, this is other command that was the only way to get the time of the program, as, was needed save or keep the information, but , the most commands return IO and not a string, so with this way, it's take the time with a string and send by parameters.

b) input () : Check what is the first command.

c) groupadd () : Add a new user to the list .

d) useradd () : Add a new user to the list and update the groups that the new user have a primary group and secondary groups.

e) showatributes () : Show all the existent user or groups in the system with their corresponding attributes.

f) finger () : show the details of a specific user account.

g) usermod () : Change the primary group or include new secondary groups to the users and update the list of groups with new groups or other associated users.

h) deluser () : Remove a user. The associated home directory (and any contents) are removed Any file or directory with this user in their ownership is go to updated as well to remove from the ownership the username. The groups will have to be updated as well.

i) delgroup () : Remove one group .The associated users is go to be modified to remove their association with the group that is being removed. Any file or directory with this group in their ownership is go to be updated as well to.

j) createdev () : create a new storage device in Megabibytes or Gigabibyes.

k) fdisk () : Show all the storage devices that exists in the system.

l) rmdev () : Remove the storage devices created in the system.

m) pvcreate () : Update the storage device and set it as managed by LVM

n) vgcreate () : Create a volume group in any storage devices.

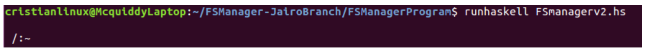
o) vgreduce () : Remove a storage device from an existent volume group.

p) vgextend () : Add a new storage device to an existent volume group.

5 User guide

This system will emulate a file system manager,The best way to get started is to download the Haskell Platform [6], that platform include the software cabal.To install the platform write the following command ,**sudo apt-get install haskell-platform**, after that update cabal with **cabal update**, and for the file system is needed the libraries time,tuple and split so to install the libraries are **install tuple, install split and install time**.

When all are installed, is needed run the file system, as figure 3.

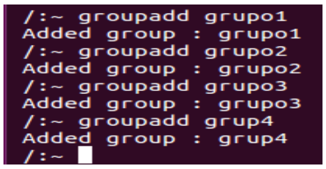


```
cristianlinux@McquiddyLaptop:~/FSManager-JairoBranch/FSManagerProgram$ runhaskell FSmanagerv2.hs
 /:~
```

Fig. 3. Running the file system.

The file system will have all the following components:
:

- **Users and Groups.**



```
/:~ groupadd grupo1
Added group : grupo1
/:~ groupadd grupo2
Added group : grupo2
/:~ groupadd grupo3
Added group : grupo3
/:~ groupadd grup4
Added group : grup4
/:~
```

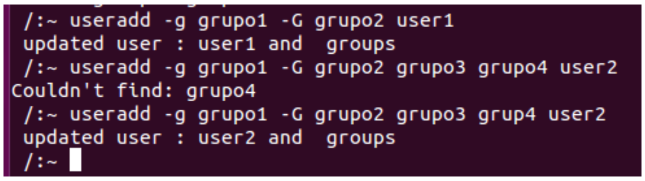
Fig. 4. Creating user groups.

As figure 4,Groupname, it can be formed by characters (a-z), numbers(0-9) and may only be up to 32 characters long.Group ID (GID),will be a number starting on 1000, It will be automatically added to the groups that are being created in increments of 1.

As figure 5,Username, it can be formed by characters (a-z), numbers(0-9) and may only be up to 32 characters long.
UserID (UID), this will be a number starting on 1000, It will be automatically added to the users that are being created in increments of 1.

Table 1. REQUIREMENTS

HARDWARE	SOFTWARE
1.3 GHz processor	Linux O.S
1GHz Ram	Haskell
256 MB Video Card	Cabal



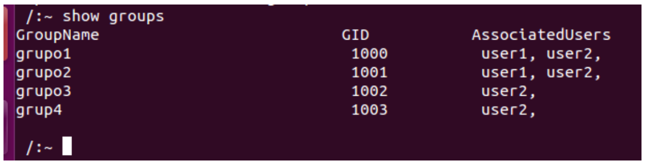
```
 /:~ useradd -g grupo1 -G grupo2 user1
 updated user : user1 and  groups
 /:~ useradd -g grupo1 -G grupo2 grupo3 grupo4 user2
Couldn't find: grupo4
 /:~ useradd -g grupo1 -G grupo2 grupo3 grup4 user2
 updated user : user2 and  groups
 /:~ 
```

Fig. 5. Creating users.

Home directory, by default it will be set to /home/¡username¿. When creating a user, this directory has to be automatically created in the system, and the ownership set to the current username and the corresponding primary group.

Associated primary group. This is a user groups that has to exists in the environment. It will be assigned to the user during the creation with -g , or with the command usermod explained below.

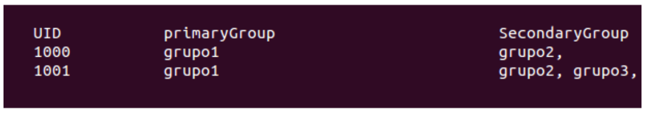
Associated secondary groups. A user can have multiple groups associated. The group(s) can be assigned to a user during its creation with the -G flag or with the command usermod explained below.



```
 /:~ show groups
GroupName                    GID          AssociatedUsers
grupo1                        1000          user1, user2,
grupo2                        1001          user1, user2,
grupo3                        1002          user2,
grup4                         1003          user2,

 /:~ 
```

Fig. 6. Show groups attributes.

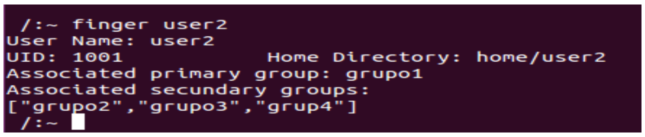
The previous command,as figure 6 will be used to list all the existent groups in the system with their corresponding attributes.



```
 UID          primaryGroup                  SecondaryGroup
 1000         grupo1                        grupo2,
 1001         grupo1                        grupo2, grupo3,
```

Fig. 7. Show users attributes.

The previous command, as figure 7 will be used to list all the existent user in the system with their corresponding attributes.

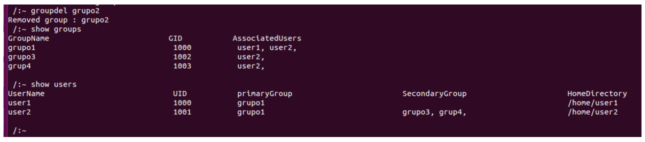


```
 /:~ finger user2
User Name: user2
UID: 1001          Home Directory: home/user2
Associated primary group: grupo1
Associated secundary groups:
["grupo2","grupo3","grup4"]
 /:~ 
```

Fig. 8. Finger command.

The previous command, as figure 8 , show the details of a specific user account.

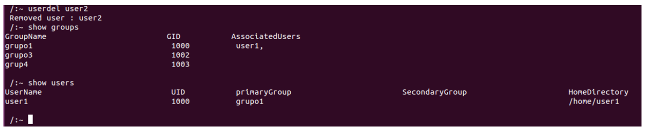
Groups can be deleted using the previous command, as figure 9. The associated users have to be modified to remove their association with the group that is



```
 /:~ groupdel grupo2
Removed group : grupo2
 /:~ show groups
GroupName                          GID          AssociatedUsers
grupo1                              1000          user1, user2,
grupo3                              1002          user2,
grup4                               1003          user2,

 /:~ show users
UserName                            UID          primaryGroup                     SecondaryGroup                  HomeDirectory
user1                               1000          grupo1                                                          /home/user1
user2                               1001          grupo1                           grupo3, grup4,                 /home/user2

 /:~
```

Fig. 9. Deleting groups.

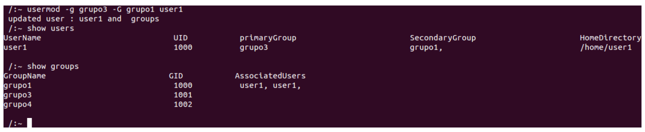
being removed. Any file or directory with this group in their ownership have to be updated as well to remove from the ownership the group.



```
 /:~ userdel user2
 Removed user : user2
 /:~ show groups
GroupName                          GID          AssociatedUsers
grupo1                              1000          user1,
grupo3                              1002
grup4                               1003

 /:~ show users
UserName                            UID          primaryGroup                     SecondaryGroup                  HomeDirectory
user1                               1000          grupo1                                                          /home/user1

 /:~ 
```

Fig. 10. Deleting users.

Users can be deleted using the previous command, as figure 10. The associated home directory (and any contents) has to be removed when the user is removed as well. Any file or directory with this user in their ownership have to be updated as well to remove from the ownership the username. The groups will have to be updated as well.

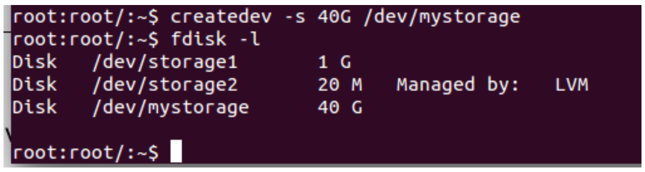


```
 /:~ usermod -g grupo3 -G grupo1 user1
 updated user : user1 and  groups
 /:~ show users
UserName                            UID          primaryGroup                     SecondaryGroup                  HomeDirectory
user1                               1000          grupo3                           grupo1,                        /home/user1

 /:~ show groups
GroupName                          GID          AssociatedUsers
grupo1                              1000          user1, user1,
grupo3                              1001
grupo4                              1002

 /:~ 
```

Fig. 11. Modifying users.

As figure 11,Users can be modified to change the primary group or to include new secondary groups.

- **Storage devices.**
  The following command , as figure 12, will create a new storage device.
  Storage devices will be created to emulate the

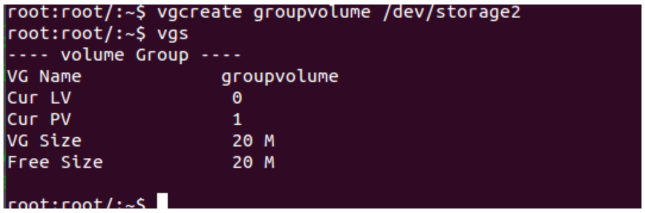


```
root:root/:~$ createdev -s 40G /dev/mystorage
root:root/:~$ fdisk -l
Disk   /dev/storage1       1 G
Disk   /dev/storage2       20 M   Managed by:   LVM
Disk   /dev/mystorage      40 G

root:root/:~$ 
```

Fig. 12. Create storage devices.

equivalent to SAN LUNs, hard drives or any other storage device required to create a file system or a logical volume.

- **Volume groups.**
  A volume group can be created with the following command, as figure 13. It will be formed by one or more storage devices.
  A volume group has the following attributes:
  Volume Group name, it can be formed by characters

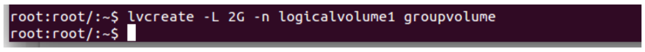


```
root:root/:~$ vgcreate groupvolume /dev/storage2
root:root/:~$ vgs
---- volume Group ----
VG Name              groupvolume
Cur LV                0
Cur PV                1
VG Size               20 M
Free Size             20 M

root:root/:~$
```

Fig. 13. Creating volume groupss.

(a-z), numbers(0-9) and may only be up to 32 characters long.
List of physical volumes associated. This is the list of storage devices that belong to this volume group.
Amount of physical volumes associated.
Amount of volumes created in the volume group.
List of volumes created in the volume group.
Total allocated size in MiB. This is the total size of the storage that has been allocated in the volume group. This is a sum of the storage devices sizes that belong to the specific volume group.
Total available size in MiB. This is the total space available to be assigned to new or existent volumes.

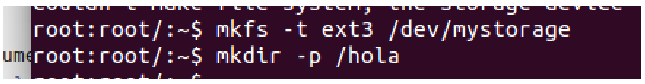
- **Logical volumes.**
  A logical volume can be created using the following command, as figure 14. It will be formed by one or more storage devices. A volume group can have multiple volumes. The size of each volume cannot exceed the total available space in the volume group at the moment of the volume creation.



```
root:root/:~$ lvcreate -L 2G -n logicalvolume1 groupvolume
root:root/:~$
```

Fig. 14. Creating a logical volume.

- **File systems.**
  The following command will be used to create a file system and a new directory . The file system type can be ext2, ext3 or ext4. The size of the file system will be inherit from the logical volume group or storage device that is being used to create the file system, as figure 15.

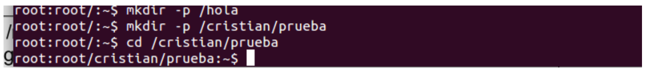


```
root:root/:~$ mkfs -t ext3 /dev/mystorage
root:root/:~$ mkdir -p /hola
```

Fig. 15. Creating a file system.

- **Files and directories.**
  All the files and directories will be created over the / (root) directory. This directory (/) cannot be used as a file system mount point of any newly created file system. It will be the main parent directory of any directory created in this system.

  The below command is going to be used to create directories. Notice the directories can have multiple levels that will be separated with the slash character (/). In the below example, a mount directory is going to be created and then inside the cristian directory another directory called prueba is going to be created. In order to create file systems of multiple levels use the mkdir command with the -p flag.
  The command **cd** will be used to navigate across all the system directories.

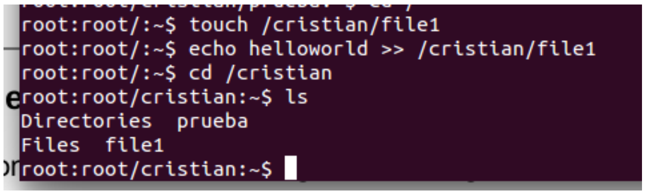


```
root:root/:~$ mkdir -p /hola
root:root/:~$ mkdir -p /cristian/prueba
root:root/:~$ cd /cristian/prueba
root:root/cristian/prueba:~$
```

Fig. 16. Creating directories and navigating directories.

  Inside a directory other directories can be created and also files. In order to create a file use the following command:

  Notice the file is created inside the /cristian directory. Other way of creating a file inside a specified directory is being moved inside that directory with the cd command.
  Once a file is created we can add contents to such file using **echo** ,as figure 17.



```
root:root/:~$ touch /cristian/file1
root:root/:~$ echo helloworld >> /cristian/file1
root:root/:~$ cd /cristian
root:root/cristian:~$ ls
Directories  prueba
Files  file1
root:root/cristian:~$
```

Fig. 17. Creating empty files and adding contents to a file.

6 Project final status

Symbolic links didn't work

File system mount points didn't work, like mount and unmount.

Remove files didn't works.

6.1 Some issues or bugs

1) Groupname and userName, it can be formed by characters (a-z), numbers(0-9), symbols and don't have a limit of characters long

2) echo only can write text to a file, without spaces.

3) The commands of files system only works without spaces.

4) T directories and files are showing unrecursyble

6.2 Challenges during the development

1) Fudgets : Couldn't find module fudgets, this took to make the interface: That was the problem when trying import the libraty fugets.

   Attemps: Download the package fugets and intall with the command cabal install in Fudgetshbcrhbd_library and after in Fudgets but did not work.
   Write the command cabal update and cabal install but did not work.With the command cabal intall Fudgets, did not work too.
   After hours doing the almost the same, it try with other library like, yamba and wxfruit, but doing the same command did not work.

2) Input Text: Design an algorithm that can take the input text.

   Solution: Write the following algorithm :
   import Control.Monad
   main =forever $ do
   putStr "# "
   l < − getLine
   putStrLn l

3) Split text :Design a algorithm that skip the spaces element from the list, when is reading the space element it continue reading the nex element.

   Attemps: Write the commands:
   split(dropBlanks & oneOf) text, but split one space and not al.
   The command dropBlanks, remove all elements of the list that was spliting.
   Other command that did not work and not was allowed: splitOn(dropblanks), split(dropBlanks On), ect.

   Solution: With the command splitOn, make a list with the words without one simbol, in this case is the space , but the problem is when are two or more spaces, because the command split one space, so in the list are spaces like elements.

4) Expected IO: When the main call a function this return other type expected, the error showed was :
   Couldn't match expected type IO t0
   with actual type String -¿ IO ()
   Probable cause: main is applied to too few arguments
   In the expression: main
   When checking the type of the IO action main

   Attemps: 1. Create the main funtion,with arguments, the arguments was sended by the fsManager funtion, who calls the main.
   2. Change the name of the main funtion,but the program show de error: Not in scope: main Perhaps you meant min (imported from Prelude)

   Solution: From the main funtion calls the fsManager with the initial arguments, and from the fsManager gets the input instructions.

7 Conclusions and Suggestions

Haskell is a strong functional paradigm, but maintaining code is very inefficient .

Almost everything are connect, so if one function changed the other function need to update.

The modules are very important ans easy to use, they give a better structure of code, when the program are running and show a error message, in the message are the name of the module or file the contain the error, so is faster to fix the problem.

To save or keep the data, this information needs to send from parameter to parameter in each function.

The command x:xs is a good instruction where x take the first element of the list xs, and the function is called recursively, the list xs, is without the first element x.So this, is a faster and easy way to take the elements of one list.

Casting from IO to String or String to IO, is impossible, the best way is that the function with IO return, take the string and print it.

8 REFERENCES

[1] "General Overview Of The Linux File System". Tldp.org. N.p., 2016. Web. 31 May 2016.

[2] Howtogeek.com. (2016). What is the Linux Kernel and What Does It Do?. [online] Available at: http://www.howtogeek.com/howto/31632/what-is-the-linux-kernel-and-what-does-it-do/ [Accessed 8 May 2016].

[3] Prof. Jennifer Vargas , 17 may 2016 , "File System Manager" Costa Rica Institute of Technology.Computer Engineering

[4] Logical Volume Management". Wikipedia. N.p., 2016. Web. 31 May 2016.

[5] 2016. [Online]. Available: https://www.sublimetext.com/3. [Accessed: 10- May- 2016].

[6] "Download Haskell Platform". Haskell.org. N.p., 2016. Web. 1 June 2016.

[7] "Chapters - Learn You A Haskell For Great Good!". Learnyouahaskell.com. N.p., 2016. Web. 1 June 2016.

[8] layout?, How. ”How To Understand The Ubuntu File System Layout?”. Askubuntu.com. N.p., 2016. Web. 1 June 2016.

[9] ”Logical Volume Management”. Wikipedia. N.p., 2016. Web. 1 June 2016.

[10] ”Logical Volume Manager (Linux)”. Wikipedia. N.p., 2016. Web. 1 June 2016.

[11] ”Chapter 5. LVM Configuration Examples”. Access.redhat.com. N.p., 2016. Web. 1 June 2016.

[12] ”Logical Volume Manager Administration”. Access.redhat.com. N.p., 2016. Web. 1 June 2016.

[13] Perez, Ivan. ”On The State Of GUI Programming In Haskell — Keera Studios”. Keera.co.uk. N.p., 2014. Web. 1 June 2016.

[14] ”Functional Reactive Programming - Haskellwiki”. Wiki.haskell.org. N.p., 2016. Web. 1 June 2016.

[15] ”Haskell Tutorial - Installation”. YouTube. N.p., 2016. Web. 1 June 2016.

[16] ”Input And Output - Learn You A Haskell For Great Good!”. Learnyouahaskell.com. N.p., 2016. Web. 1 June 2016.

[17] Haskell?, How. ”How To Split A String In Haskell?”. Stackoverflow.com. N.p., 2016. Web. 1 June 2016.

[18] ”How To Work On Lists - Haskellwiki”. Wiki.haskell.org. N.p., 2016. Web. 1 June 2016.

THE AUTHORS

Table 2. Details of the activities performed by Cristian Roberto Castillo Mcquiddy

Date.	Activity	Description	Hours
17/05/2016	Introduccion Haskell	Research about what's Haskell, how works the functions, list and tuples. [7]	2
18/05/2016	Haskell Types	Type variables, type classes ans expresion in haskell. [7]	2
18/05/2016	Haskell Recursion	Learning about , A few more recursive functions, algorithm quick sort,Maps and filters, Lambdas, Function application with $ [7]	2
19/05/2016	How file system works	Reading the proyect especfications. Learning how fily system in linux works and, like: LVM, logical volumen, physicals volumen. [8] [9] [10] [11].	4
20/05/2016	GUI HASKELL Research	There are GUI libraries IMPERACTIVES likes QT,WX,Gtk+ and FUNTIONAL likes Fudgets and Functional Reactive Programming (FRP) the FUNTIONAL libraries are perfect with this proyect, because it can not use variables . [12] [13]	1
20/05/2016	Testing Fudgets	Reading the manual and installing fudgets . [14].	4
22/05/2016	Input Text	Reading wich command can will use and a algorithm that can take the input text. [15]	2
22/05/2016	Split text	Design a algorithm that skip the spaces element from the list, when is reading the space element it continue reading the nex element. [16]	4
22/05/2016	How work a list	Design algorithm that allowe the estructure of new Group and users [17]	2
23/05/2016	Groups and users	Documenting code.The funtion add group and user.	1
24/05/2016	How work a list	Merge the tuples modules made by jairo , the program that check the information or data . [18]	3
24/05/2016	Users and Groups	Creating users and groups	6
25/05/2016	Users and Groups	Design an algorithm that update the user and groups list	3
25/05/2016	Input Text and groups	Remove the spaces of the text an add secondary groups in users	
25/05/2016	Symbols input	Detecting and remove some symbols	1
		Total amount of hours:	40

Table 3. Details of the activities performed by Jairo Mndez Martnez

	Activity	Description
	Investigation about Haskell	Research on what is haskell, begins by following a tutorial and initiating its codification.
	Continue with the investigation about haskell	Begins the basic approach of the project, raising the structure and requirements for its development and continuing a research on basic functions and functionality haskell lists
	Investigation about how lists and tuples works in Haskell	continuing research on basic functions and functionality haskell lists, and how to create my own data and my own structure data (a simple linked list) and how to get an element from a list and tuple
	Inestigation about modules in Haskell	Finish tha simple linked list, and search how to create a module in haskell and imported from another file
	Starting to studying and understending the requirements of the proyect	Generating a table with all the requirements and problems, try
	Starting to work with groups and users	Startig to generate the logical of the users and groups, groupadd and useradd
	Starting with the management of storage devices	Generate the structure of a storage device with a list of tuples, where the information of the device is storage
	Working with volumes	Implementation of tha functions that will described the functionallity of them
16	Working on mkdir and touch functions	How to generate directories, how to access to them and how to generate files
	Working with files and directories	Implementing the logical of the remove, chmod, echo and ls of the files and directories
		Total amount of hours: