**Unix File System Project Tasks**

1. To decide on how to store files in DB
   1. Should we have to store data in a separate table based on file size
      1. Experiment storing file in a single blob and do search for string
      2. Experiment storing file into multiple records partitioned based on block size and do search
         1. In this case, there is possibility that some string will be split and it will be stored in different records. Hence grep for such strings shall consider adjacent record too.

1. Folder and files parsing
   1. Should we no need to worry about hidden files as we need to support only “ls -l” no need to support “ls -al” ?
   2. ls command and cd command with tab press, displays different results. Cd displays hidden folders (after pressing TAB twice)

mk@mkserver:~$ ls

chesstest mysql-apt-config\_0.8.14-1\_all.deb project.pdf sample.out sample.sql temp temp.txt test testing testing1

mk@mkserver:~$ cd

.cache/ chesstest/ .gnupg/ .ssh/ temp/ test/ testing/ testing1/ .vim/

* 1. Challenges on soft link and hard link
     1. **TBD**

1. Project Structure
   1. rdbsh.py
      1. It’s the main bash file, it shall have API implementation of all commands supported (may be a switch case with currently supported commands and its functionality)
   2. mysqlfscreate.py
      1. create DB from FS path (provided by rdbsh.py) by calling MYSQLEXEC APIs in mysqlfsmgr.py
   3. mysqlfsmgr.py
      1. Implement necessary functionalities to connect to mysql
      2. Implement functionality to execute mysql command (MYSQLEXEC)
      3. Implement query functionality for each command (cd , ls, find, grep)
2. Python file execution (rdbsh.py)
   1. Command line argument
      1. Mysql user name : -u (Eg : -u DBA)
      2. Mysql password : -p (Eg: -p PASSWORD)
      3. Use existing FS database name : -e (Eg: -e FILESYSTEMDB)
      4. Create new DB : -c (Eg: -c FILESYSTEMDB)
      5. FS Source path : -f (-f / or -f /temp/temp1)
   2. Once parameter validation and DB creation is successful, directly go to the terminal with the root folder (unixfs:/$). [ADV : If we have to display respective user, we may have to parse /etc/passwd file and find out user name which has /home with /bin/bash.]
   3. ADV : How about showing FS loading progress? – For this, we might have to do one round of finding out all files and folders. We can create master table.
   4. What if, if DB already exists?, lets do these checks if we have time.
3. Shell:
   1. We should issue query internally to find out the entries of the current directory
   2. We can also support “pwd” command

mk@mkserver:~$ pwd

/home/mk

* 1. Group them by directory and executables.
     1. Grouping by directory will help in CD command
     2. Grouping by executables will help in ./ command

1. TAB press in a prompt : when entering a string and then press TAB more than once, it should display all executables starting with that string name
2. PATH Variable
   1. In Linux, PATH variable is below.

mk@mkserver:~/temp$ echo $PATH

/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin

* 1. Requirement here is to execute any executable program in the PATH
  2. It is better that the shell program shall CACHE the executable file names along with PATH so that when user enters EXE name and press enter, we shall be able to quickly find the related FOLDER, download EXE from DB and execute it.
  3. Will there could be two executables with same name in different paths??
  4. Before downloading an EXE, the previous EXE shall be deleted and new EXE shall be downloaded from DB
  5. Need to FIND relevant python way of executing the exe (Subprocess?)
  6. Its better to create a local directory (fstemp) to download the EXE for execution

1. CD command
   1. Add basic CD “folder\_name”
   2. Supporting TAB key press with “CD <blank>“ : for every tab press > 1, display the contents the directory
   3. Supporting TAB key press with “CD <partial\_folder\_name>” : For single TAB press, display the common strings and then display the folders.

mk@mkserver:~$ cd t  cd te (while pressing TAB)

temp/ test/ testing/ testing1/

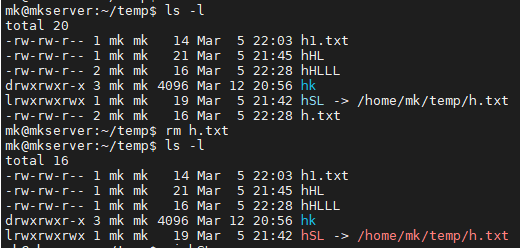
* 1. Once CD command is successful, we shall do as mentioned in section 5) Shell.
  2. Should handle CD . and CD .. command. Should we store the previous folder information (using fid) or get it again from DB?.
  3. ADV : mk@mkserver:~/temp$ cd ../..
  4. ADV : mk@mkserver:~/temp$ cd ../home

1. LS command
   1. LS command displays files in number => alphabetical order
   2. ADV : Color coding for files and folder categories

mk@mkserver:~$ ls

1.txt chesstest mysql-apt-config\_0.8.14-1\_all.deb project.pdf sample.out sample.sql temp temp.txt test testing testing1

* 1. For ls -l, need to retrieve the information from properties table and display them
  2. For soft links, need to mention where it points too



* 1. For hard links, only the nlink will be more than 1

1. FIND Command  search for files in a directory hierarchy
   1. Simple “FIND” and then enter, it displays all contents in current directory

mk@mkserver:~/temp$ find

.

./hSL

./hHL

./hHol

./hk

./hk/yy

./hHl

./h1.txt

./hHLLL

mk@mkserver:~/temp$

* 1. FIND with folder path

mk@mkserver:~/temp$ find /home/mk/temp/

/home/mk/temp/

/home/mk/temp/hSL

/home/mk/temp/hHL

/home/mk/temp/hHol

/home/mk/temp/hk

/home/mk/temp/hk/yy

/home/mk/temp/hHl

/home/mk/temp/h1.txt

/home/mk/temp/hHLLL

mk@mkserver:~/temp$

* 1. Find with sub-directories in search path

mk@mkserver:~/temp$ ls

b.txt h1.txt hHl hHL hHLLL hHol hk ZSL

mk@mkserver:~/temp$ find

.

./hHL

./hHol

./hk

./hk/yy

./hk/yy/d.txt

./hk/c.txt

./hHl

./ZSL

./b.txt

./h1.txt

./hHLLL

mk@mkserver:~/temp$ find /home/mk/te

temp/ test/ testing/ testing1/

mk@mkserver:~/temp$ find /home/mk/temp/

/home/mk/temp/

/home/mk/temp/hHL

/home/mk/temp/hHol

/home/mk/temp/hk

/home/mk/temp/hk/yy

/home/mk/temp/hk/yy/d.txt

/home/mk/temp/hk/c.txt

/home/mk/temp/hHl

/home/mk/temp/ZSL

/home/mk/temp/b.txt

/home/mk/temp/h1.txt

/home/mk/temp/hHLLL

mk@mkserver:~/temp$

1. GREP Command
   1. Sample

mk@mkserver:~/temp$ grep mk /etc/passwd

mk:x:1000:1000:smkumar:/home/mk:/bin/bash

mk@mkserver:~/temp$ grep He c.txt

Hello

mk@mkserver:~/temp$ grep "Hel" c.txt

Hello

mk@mkserver:~/temp$ grep 'Hel' c.txt

Hello

mk@mkserver:~/temp$ grep He- c.txt

mk@mkserver:~/temp$ grep Helo c.txt

mk@mkserver:~/temp$ grep Hel d.txt c.txt

d.txt:Hello

c.txt:Hello

mk@mkserver:~/temp$ grep -in Hello c.txt  option -n for line number, i for ignore case

1:Hello

mk@mkserver:~/temp$ grep -n Hello c.txt

1:Hello

mk@mkserver:~/temp$ grep -n kumar b.txt

1:Hello kumar

2:kumar

4:kumar

mk@mkserver:~/temp$ grep -n Hello kumar b.txt

grep: kumar: No such file or directory

b.txt:1:Hello kumar

mk@mkserver:~/temp$ grep -n 'Hello kumar' b.txt

1:Hello kumar

**ISSUES:**

**Creation of database:**

**1)** The fileswap.img of size 2 GB is deleted from the original file structure. The reason is that mysql server connection is getting lost while reading that file even chunk by chunk.

2) Currently the variable “max\_allowed\_packet” is set to its maximum limit(1 GB) at the mysql server. Default value of max\_allowed\_packet is 64 MB. Because there is a file initrd.img.old of 57 MB which is difficult to read and throwing error.

3) While creating database, the python file hangs at a folder initctl and some other points where we have to issue the command Ctrl + C to continue further and then gets completed.

4) Create a separate table for links which will have columns(fileid --→ targetfileid)

5) Now os.scan\_dir() scans hidden files as well.