**Programming assignment 2**

**CS-465 Introduction to Cybersecurity**

**Audunn Alexandersson**

**Login ID: aha0026**

**WVU ID: 800-267530**

**4/17/2018**

**Program status:** The program should be fully functional and working.

**Environment description:**

* This project is coded in Python 3 with the PyCharm from JetBrains IDE.
* I used Cmd module and DefaultDict libraries for this project.
* System this project was programmed on is a mid 2015 model MacBook Pro 15” with 2,5 GHz Intel Core i7 processor and 16 GB 1600 MHz DDR3 RAM.

**Description of Accounts.txt**

I did not do the bonus part of this assignment, therefore my Accounts.txt is not complicated. I used dictionary to keep track of accounts within the running project and after calling end I then uploaded every username and password straight up to the text file with pain passwords which is not good for keeping such secrets.

**Test case 1:**

useradd root password321

login root password321

mkfile presents.txt

mkfile teeth.txt

useradd santa christmas

useradd toothfairy brushTeeth2timesAday

groupadd goodkids

groupadd badkids

chown goodkids santa

chown sadkids toothfairy

chmod badkids rwx rwx rwx

chmod good rwx --- rwx

logout

login santa christmas

read goodkids

read badkids

execute goodkids

mkfile needtobuylist.txt

execute needtobuylist.txt

logout

end

**Final Content of User Created Files**

Needtobuylist.txt  
Presents.txt

Teeth.txt

**Groups.txt content**

goodkids:

badkids:

**Files.txt content**

presents.txt: root nil rw- --- ---

teeth.txt: root nil rw- --- ---

needtobuylist.txt: santa nil rw- --- ---

**Accounts.txt content**

root password321

santa christmas

toothfairy brushTeeth2timesAday

**Test case 2:**

useradd root rooter

login root rooter

useradd joe water123

useradd jessie bratz321

groupadd besties

usergrp joe best

usergrp joe besties

mkfile friends.txt

chmod friends.txt rw- rwx ---

logout

login joe water

login joe water123

chown friends.txt jessie

read friends.txt

ls friends.txt

logout

login jessie bratz321

mkfile tomthebest.txt

write tomthebest.txt I am in love with Tom

read tomthebest.txt

logout

login root rooter

chmod tomthebest.txt rwx rwx rwx

logout

end

**Final Content of User Created Files**

tomthebest.txt – “I am in love with Tom”  
friends.txt

**Groups.txt content**

besties: joe

**Files.txt content**

friends.txt: root nil rw- rwx ---

tomthebest.txt: jessie nil rwx rwx rwx

**Accounts.txt content**

root rooter

joe water123

jessie bratz321

**CODE**

import cmd  
from collections import defaultdict  
  
ACCOUNTS\_FILE = "accounts.txt"  
AUDIT\_FILE = "audit.txt"  
GROUPS\_FILE = "groups.txt"  
FILES\_FILE = "files.txt"  
  
CURRENT\_USER = ""  
USERLIST = dict()  
USERGROUPS = dict()  
USERFILES = defaultdict(list)  
  
AUDIT = open(AUDIT\_FILE, 'w')  
  
#Updates the file.txt file  
def update\_file():  
 with open(FILES\_FILE, 'w+') as f:  
 f.seek(0)  
 for key, item in USERFILES.items():  
 info = key + ': ' + " ".join(item) + "\n"  
 f.write(info)  
  
#Updates the groups.txt file  
def update\_group():  
 with open(GROUPS\_FILE, 'w+') as g:  
 g.seek(0)  
 for key,item in USERGROUPS.items():  
 info = key + ': ' + item + "\n"  
 g.write(info)  
  
#Updates the account.txt file  
def update\_account():  
 with open(ACCOUNTS\_FILE, 'w+') as a:  
 a.seek(0)  
 for key, item in USERLIST.items():  
 info = key + " " + item + "\n"  
 a.write(info)  
  
#Takes in the username and password parameters and passes it into global userlist dict.  
def useradd(username, password):  
 USERLIST[username] = password  
 print("User %s created" % username)  
 AUDIT.write("User %s created\n" % username)  
  
  
#Takes in username and password parameters and checks if the username and password are correct  
#then makes the Current user variable be the correct logged in user.  
def login(username, password):  
 if username in USERLIST:  
 if USERLIST[username] == password:  
 global CURRENT\_USER  
 CURRENT\_USER = username  
 print("User %s logged in" % username)  
 AUDIT.write("User %s logged in\n" % username)  
 else:  
 print("Login failed: invalid username or password")  
 AUDIT.write("Login failed: invalid username or password\n")  
 else:  
 print("Login failed: invalid username or password")  
 AUDIT.write("Login failed: invalid username or password\n")  
  
#Takes the global current user variable and logs the user out by making it be a empty string  
def logout():  
 global CURRENT\_USER  
 print("User %s logged out" % CURRENT\_USER)  
 AUDIT.write("User %s logged out\n" % CURRENT\_USER)  
 CURRENT\_USER = ""  
  
  
#Creates a group by taking a groupname and matching it with the Usergroup global dict.  
def groupadd(groupname):  
 if groupname not in USERGROUPS:  
 if groupname.lower() != 'nil':  
 USERGROUPS[groupname] = ''  
 print("Group %s created" % groupname)  
 AUDIT.write("Group %s created\n" % groupname)  
 else:  
 print("Error: group name cannot be nil")  
 AUDIT.write("Error: group name cannot be nil\n")  
 else:  
 print("Error: group %s already exists" % groupname)  
 AUDIT.write("Error: group %s already exists\n" % groupname)  
  
  
#Adds the spesific user to the group  
def usergrp(username, groupname):  
 s = USERGROUPS[groupname]  
 s += username + " "  
 USERGROUPS[groupname] = s  
 print("User %s added to group %s" % (username, groupname))  
 AUDIT.write("User %s added to group %s\n" % (username, groupname))  
  
  
#Creates a new file with the default permissions and currently logged in user.  
def mkfile(filename):  
 if check\_filename(filename) == False:  
 print("Error: filename cannot be %s" % (filename))  
 AUDIT.write("Error: filename cannot be %s" % (filename))  
 return  
  
 if filename not in USERFILES:  
 #user, group, owner perm, group perm, other perm.  
 fileinfo = [CURRENT\_USER, 'nil', 'rw-', '---', '---']  
 USERFILES[filename] = fileinfo  
 #to create the file in the folder  
 with open(filename, "w+") as a:  
 pass  
 print("File %s with owner %s and default permissions created" % (filename, CURRENT\_USER))  
 AUDIT.write("File %s with owner %s and default permissions created\n" % (filename, CURRENT\_USER))  
 else:  
 print("Error: file %s already exists" % filename)  
 AUDIT.write("Error: file %s already exists\n" % filename)  
  
  
#updates the permissions for each group  
def chmod(filename, owner, group, other):  
 if check\_filename(filename) == False:  
 print("Error: filename cannot be %s" % (filename))  
 AUDIT.write("Error: filename cannot be %s" % (filename))  
 return  
  
 if filename in USERFILES:  
 if USERFILES[filename][0] == CURRENT\_USER or CURRENT\_USER == 'root':  
 USERFILES[filename][2] = owner  
 USERFILES[filename][3] = group  
 USERFILES[filename][4] = other  
 print("Permissions for %s set to %s %s %s by %s" % (filename, owner, group, other, CURRENT\_USER))  
 AUDIT.write("Permissions for %s set to %s %s %s by %s\n" % (filename, owner, group, other, CURRENT\_USER))  
 else:  
 print("Error with chmod: %s does not have the permission to change %s" % (CURRENT\_USER, filename))  
 AUDIT.write("Error with chmod: %s does not have the permission to change %s\n" % (CURRENT\_USER, filename))  
 else:  
 print("Error with chmod: file %s not found" % (filename))  
 AUDIT.write("Error with chmod: file %s not found\n" % (filename))  
  
  
#Changes the owner of the file  
def chown(filename, username):  
 if check\_filename(filename) == False:  
 print("Error: filename cannot be %s" % (filename))  
 AUDIT.write("Error: filename cannot be %s" % (filename))  
 return  
  
 if filename in USERFILES:  
 USERFILES[filename][0] = username  
 print("Owner of %s changed to %s" % (filename, username))  
 AUDIT.write("Owner of %s changed to %s\n" % (filename, username))  
 else:  
 print("Error with chown: file %s not found" % (filename))  
 AUDIT.write("Error with chown: file %s not found\n" % (filename))  
  
#changes the groupname to something else if he has a permission to do that  
def chgrp(filename, groupname):  
 if check\_filename(filename) == False:  
 print("Error: filename cannot be %s" % (filename))  
 AUDIT.write("Error: filename cannot be %s" % (filename))  
 return  
  
 if filename in USERFILES:  
 if USERFILES[filename][1] == "nil":  
 if CURRENT\_USER == USERFILES[filename][0] or CURRENT\_USER == "root":  
 if CURRENT\_USER in USERGROUPS[groupname] or CURRENT\_USER == "root":  
 USERFILES[filename][1] = groupname  
 print("Group for %s changed to %s by %s" % (filename, groupname, CURRENT\_USER))  
 AUDIT.write("Group for %s changed to %s by %s\n" % (filename, groupname, CURRENT\_USER))  
 else:  
 print("Error with chgrp: User %s is not a member of group %s" % (CURRENT\_USER, groupname))  
 AUDIT.write("Error with chgrp: User %s is not a member of group %s\n" % (CURRENT\_USER, groupname))  
 else:  
 print("Error with chgrp: User %s does not have the permission to change %s" % (CURRENT\_USER, filename))  
 AUDIT.write("Error with chgrp: User %s does not have the permission to change %s\n" % (CURRENT\_USER, filename))  
 else:  
 print("Error: file %s already belongs to a group" % filename)  
 AUDIT.write("Error: file %s already belongs to a group\n" % filename)  
 else:  
 print("Error with chgrp: file %s not found" % (filename))  
 AUDIT.write("Error with chgrp: file %s not found\n" % (filename))  
  
  
#helper function to read in the text of the file  
def read\_text(filename):  
 with open(filename, 'r') as f:  
 content = f.read()  
 print("User %s reads %s as: %s" % (CURRENT\_USER, filename, content))  
 AUDIT.write("User %s reads %s as:\n %s" % (CURRENT\_USER, filename, content))  
  
#helper function to write a text to a file  
def write\_text(filename,text):  
 text = " ".join(text)  
 text += '\n'  
 with open(filename, 'a') as f:  
 f.write(text)  
 print("User %s wrote to %s: %s" % (CURRENT\_USER, filename, text))  
 AUDIT.write("User %s wrote to %s: %s" % (CURRENT\_USER, filename, text))  
  
  
#reads in information if the correct permission is allowed  
def read(filename):  
 if check\_filename(filename) == False:  
 print("Error: filename cannot be %s" % (filename))  
 AUDIT.write("Error: filename cannot be %s" % (filename))  
 return  
  
 if filename in USERFILES:  
 file = USERFILES[filename]  
 #owner  
 if CURRENT\_USER == file[0] and file[2][0] == 'r':  
 read\_text(filename)  
 #group  
 elif file[1] != 'nil' and CURRENT\_USER in USERGROUPS[file[1]] and file[3][0] == 'r' and CURRENT\_USER != file[0]:  
 read\_text(filename)  
 #other  
 elif file[4][0] == 'r' and CURRENT\_USER != file[0] and CURRENT\_USER not in USERGROUPS[file[1]]:  
 read\_text(filename)  
 else:  
 print("User %s denied read access to %s" % (CURRENT\_USER, filename))  
 AUDIT.write("User %s denied read access to %s\n" % (CURRENT\_USER, filename))  
 else:  
 print("Error with read: file %s not found" % (filename))  
 AUDIT.write("Error with read: file %s not found\n" % (filename))  
  
#Writes out text if the correct permission is allowed  
def write(filename, text):  
 if check\_filename(filename) == False:  
 print("Error: filename cannot be %s" % (filename))  
 AUDIT.write("Error: filename cannot be %s" % (filename))  
 return  
  
 if filename in USERFILES:  
 file = USERFILES[filename]  
 #owner  
 if CURRENT\_USER == file[0] and file[2][1] == 'w':  
 write\_text(filename, text)  
 #group  
 elif file[1] != 'nil' and CURRENT\_USER in USERGROUPS[file[1]] and file[3][1] == 'w' and CURRENT\_USER != file[0]:  
 write\_text(filename, text)  
 #other  
 elif file[4][1] == 'w' and CURRENT\_USER != file[0] and CURRENT\_USER not in USERGROUPS[file[1]]:  
 write\_text(filename, text)  
 else:  
 print("User %s denied write access to %s" % (CURRENT\_USER, filename))  
 AUDIT.write("User %s denied write access to %s\n" % (CURRENT\_USER, filename))  
 else:  
 print("Error with write: file %s not found" % (filename))  
 AUDIT.write("Error with write: file %s not found\n" % (filename))  
  
#checks the permission to execute a file  
def execute(filename):  
 if check\_filename(filename) == False:  
 print("Error: filename cannot be %s" % (filename))  
 AUDIT.write("Error: filename cannot be %s" % (filename))  
 return  
  
 if filename in USERFILES:  
 file = USERFILES[filename]  
 #owner  
 if CURRENT\_USER == file[0] and file[2][2] == 'x':  
 print("File %s executed by %s" % (filename, CURRENT\_USER))  
 AUDIT.write("File %s executed by %s\n" % (filename, CURRENT\_USER))  
 #group  
 elif file[1] != 'nil' and CURRENT\_USER in USERGROUPS[file[1]] and file[3][2] == 'x' and CURRENT\_USER != file[0]:  
 print("File %s executed by %s" % (filename, CURRENT\_USER))  
 AUDIT.write("File %s executed by %s\n" % (filename, CURRENT\_USER))  
 #other  
 elif file[4][2] == 'x' and CURRENT\_USER != file[0] and CURRENT\_USER not in USERGROUPS[file[1]]:  
 print("File %s executed by %s" % (filename, CURRENT\_USER))  
 AUDIT.write("File %s executed by %s\n" % (filename, CURRENT\_USER))  
 else:  
 print("User %s denied execute access to %s" % (CURRENT\_USER, filename))  
 AUDIT.write("User %s denied execute access to %s\n" % (CURRENT\_USER, filename))  
 else:  
 print("Error with execute: file %s not found" % (filename))  
 AUDIT.write("Error with execute: file %s not found\n" % (filename))  
  
#shows the listing of a file and it's permissions  
def ls(filename):  
 if check\_filename(filename) == False:  
 print("Error: filename cannot be %s" % (filename))  
 AUDIT.write("Error: filename cannot be %s" % (filename))  
 return  
  
 if filename in USERFILES:  
 pri = filename  
 for i in USERFILES[filename]:  
 pri += " " + i  
 print(pri)  
 AUDIT.write(pri + "\n")  
 else:  
 print("Error with ls: file %s not found" % (filename))  
 AUDIT.write("Error with ls: file %s not found\n" % (filename))  
  
#ends the process and uploads data into files.  
def end():  
 AUDIT.close()  
 update\_account()  
 update\_file()  
 update\_group()  
  
def check\_filename(filename):  
 if filename == ACCOUNTS\_FILE or filename == GROUPS\_FILE or filename == AUDIT\_FILE or filename == FILES\_FILE:  
 return False  
 return True  
  
#command line helper  
class Program(cmd.Cmd):  
  
 prompt = ""  
  
 def do\_useradd(self, arg):  
 user = arg.split()  
 if CURRENT\_USER == '' or CURRENT\_USER == 'root':  
 if user[0] not in USERLIST:  
 useradd(user[0], user[1])  
 else:  
 print("Error: user %s already exists" % user[0])  
 AUDIT.write("Error: user %s already exists\n" % user[0])  
 else:  
 print("Error: only root may issue useradd command")  
 AUDIT.write("Error: only root may issue useradd command\n")  
  
 def do\_login(self, arg):  
 user = arg.split()  
 if CURRENT\_USER == "":  
 login(user[0], user[1])  
 else:  
 print("Login failed: simultaneous login not permitted")  
 AUDIT.write("Login failed: simultaneous login not permitted\n")  
  
 def do\_logout(self, arg):  
 if CURRENT\_USER != "":  
 logout()  
 else:  
 print("Logout failed: must be logged in to logout")  
 AUDIT.write("Logout failed: must be logged in to logout\n")  
  
 def do\_groupadd(self, arg):  
 if CURRENT\_USER == 'root':  
 groupadd(arg)  
 else:  
 print("Error: only root may issue groupadd command")  
 AUDIT.write("Error: only root may issue groupadd command\n")  
  
 def do\_usergrp(self, arg):  
 userg = arg.split()  
 if CURRENT\_USER == "root":  
 if userg[1] in USERGROUPS:  
 usergrp(userg[0], userg[1])  
 else:  
 print("Error: %s does not exist" % userg[1])  
 AUDIT.write("Error: %s does not exist\n" % userg[1])  
 else:  
 print("Error: only root may issue usergrp command")  
 AUDIT.write("Error: only root may issue usergrp command\n")  
  
 def do\_mkfile(self, arg):  
 if CURRENT\_USER != "":  
 mkfile(arg)  
 else:  
 print("Error: user must be logged in to issue mkfile command")  
 AUDIT.write("Error: user must be logged in to issue mkfile command\n")  
  
 def do\_chmod(self, arg):  
 com = arg.split()  
 if(CURRENT\_USER != ""):  
 chmod(com[0], com[1], com[2], com[3])  
 else:  
 print("Error: user must be logged in to issue chmod command")  
 AUDIT.write("Error: user must be logged in to issue chmod command\n")  
  
 def do\_chown(self, arg):  
 com = arg.split()  
 if CURRENT\_USER == "root":  
 chown(com[0], com[1])  
 else:  
 print("Error: only root may issue chown command")  
 AUDIT.write("Error: only root may issue chown command\n")  
  
 def do\_chgrp(self, arg):  
 com = arg.split()  
 if CURRENT\_USER != "":  
 chgrp(com[0], com[1])  
 else:  
 print("Error: user must be logged in to issue chgrp command")  
 AUDIT.write("Error: user must be logged in to issue chgrp command\n")  
  
 def do\_read(self, arg):  
 if CURRENT\_USER != "":  
 read(arg)  
 else:  
 print("Error: user must be logged in to issue read command")  
 AUDIT.write("Error: user must be logged in to issue read command\n")  
  
 def do\_write(self, arg):  
 com = arg.split()  
 if CURRENT\_USER != "":  
 write(com[0], com[1:])  
 else:  
 print("Error: user must be logged in to issue write command")  
 AUDIT.write("Error: user must be logged in to issue write command\n")  
  
 def do\_execute(self, arg):  
 if CURRENT\_USER != "":  
 execute(arg)  
 else:  
 print("Error: user must be logged in to issue execute command")  
 AUDIT.write("Error: user must be logged in to issue execute command\n")  
  
 def do\_ls(self, arg):  
 if CURRENT\_USER != "":  
 ls(arg)  
 else:  
 print("Error: user must be logged in to issue ls command")  
 AUDIT.write("Error: user must be logged in to issue ls command\n")  
  
 def do\_end(self, line):  
 end()  
 return True  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 prog = Program()  
 prog.cmdloop()