Alexis Navarro
88757719

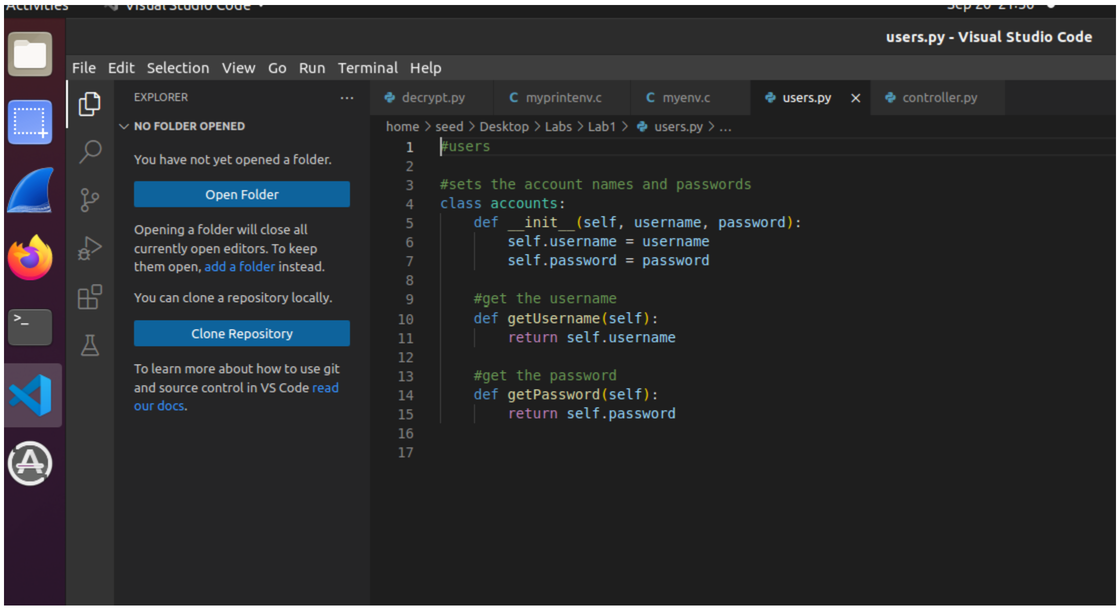
Question 1: Please answer the following questions.

a. **(10 points)** List all the attributes that are stored for an individual user in a shadow file.
    a. User login name
    b. id (Id of algorithm used)
    c. $salt$ (Random data making passwords unique)
    d. $hash$ (Hash of the user password)
    e. Days since epoch of last password change
    f. Minimum days until password change allowed
    g. Maximum days until password change allowed
    h. Maximum days of password is valid
    i. Number of days before warning for expiration
    j. Number of days without login after expiration before account locked
    k. Number of days since account was locked

b. **(5 points)** Where do the passwords in Windows system get stored?
    a. In the credential manager

Question 2: Please perform the following tasks.

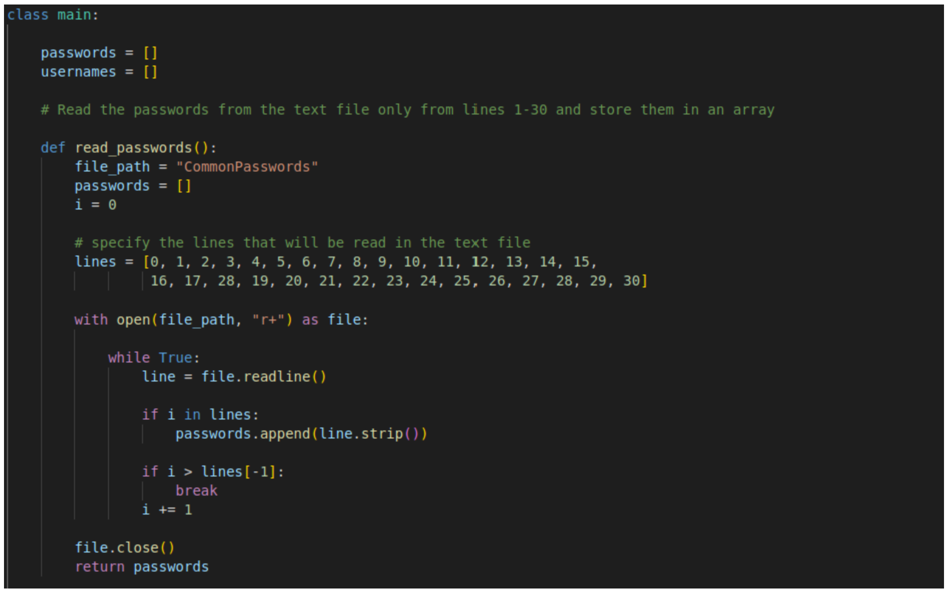
a. **(10 points)** Write a well-documented Python program that creates 30 users and uses passwords from the commonPasswdFile.txt (provided in the Assignment 1 folder).


users.py - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder
Opening a folder will close all currently open editors. To keep them open, add a folder instead.
You can clone a repository locally.
Clone Repository
To learn more about how to use git and source control in VS Code read our docs.
decrypt.py
myprintenv.c
myenv.c
users.py
controller.py
home > seed > Desktop > Labs > Lab1 > users.py > ...

```
#users

#sets the account names and passwords
class accounts:
    def __init__(self, username, password):
        self.username = username
        self.password = password

    #get the username
    def getUsername(self):
        return self.username

    #get the password
    def getPassword(self):
        return self.password
```

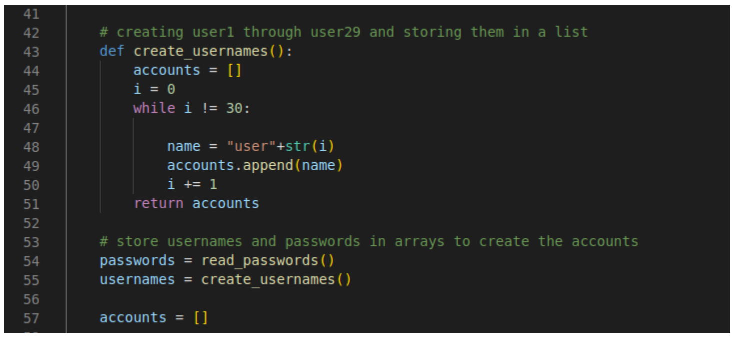
This is just a section of my code in which I am able to actually add the users into the linux vm. I started off by creating a users.py file which will construct the users which holds the username and password once it has been obtained by the text file given. Reason to this is just to be able to access the username and password of the accounts easily since I will always have the 2 values separated.

Alexis Navarro
88757719



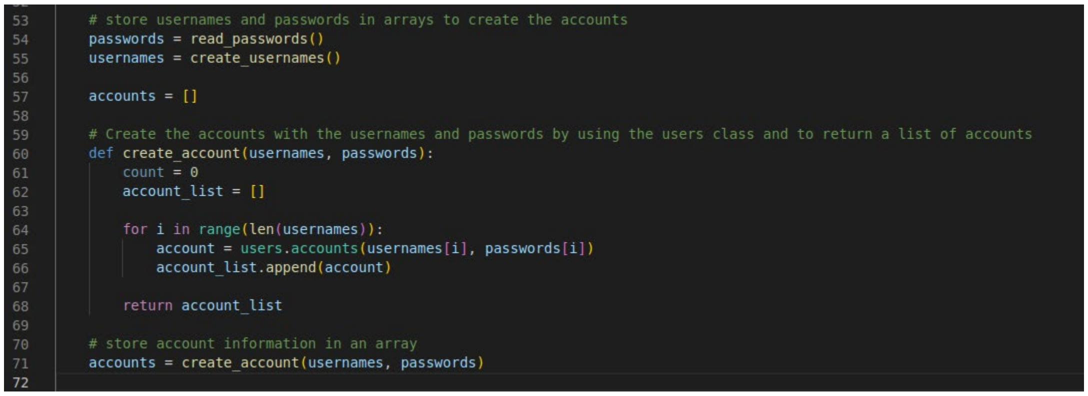
```python
class main:

    passwords = []
    usernames = []

    # Read the passwords from the text file only from lines 1-30 and store them in an array

    def read_passwords():
        file_path = "CommonPasswords"
        passwords = []
        i = 0

        # specify the lines that will be read in the text file
        lines = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
                 16, 17, 28, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]

        with open(file_path, "r+") as file:

            while True:
                line = file.readline()

                if i in lines:
                    passwords.append(line.strip())

                if i > lines[-1]:
                    break
                i += 1

        file.close()
        return passwords
```

Here we can see the read method that will be reading the text file, only that here we just care about lines 1 to 30 for the passwords, hence having a list variable specifying the lines we want to read later on. Once the text file is open, we will iterate through the file for 30 times and strip the "\n" at the end of every string and then append it to a list. If we ever go over the amount of lines we want then we end the reading, but luckly we obtain the 30 passwords.



```python

    # creating user1 through user29 and storing them in a list
    def create_usernames():
        accounts = []
        i = 0
        while i != 30:

            name = "user"+str(i)
            accounts.append(name)
            i += 1
        return accounts

    # store usernames and passwords in arrays to create the accounts
    passwords = read_passwords()
    usernames = create_usernames()

    accounts = []
```

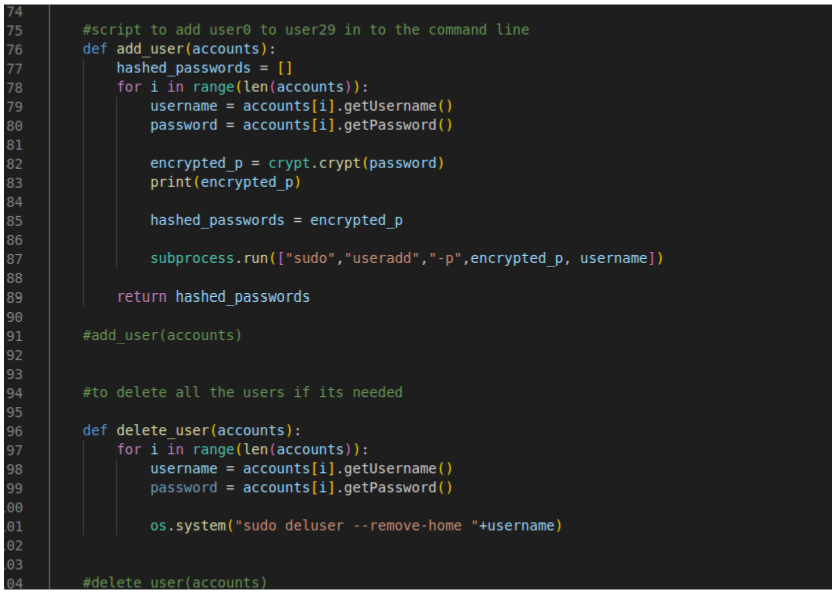
Here we are just making the usernames which will just be user0 up to user 29 for the total 30 users that will use the 30 passwords.



```python
    # store usernames and passwords in arrays to create the accounts
    passwords = read_passwords()
    usernames = create_usernames()

    accounts = []

    # Create the accounts with the usernames and passwords by using the users class and to return a list of accounts
    def create_account(usernames, passwords):
        count = 0
        account_list = []

        for i in range(len(usernames)):
            account = users.accounts(usernames[i], passwords[i])
            account_list.append(account)

        return account_list

    # store account information in an array
    accounts = create_account(usernames, passwords)

```

Alexis Navarro
88757719

Here we are now able to create the accounts as objects, not as users in the linux machine, for ease I just created all the accounts with their respective user and password then stored them in a list to use locally.

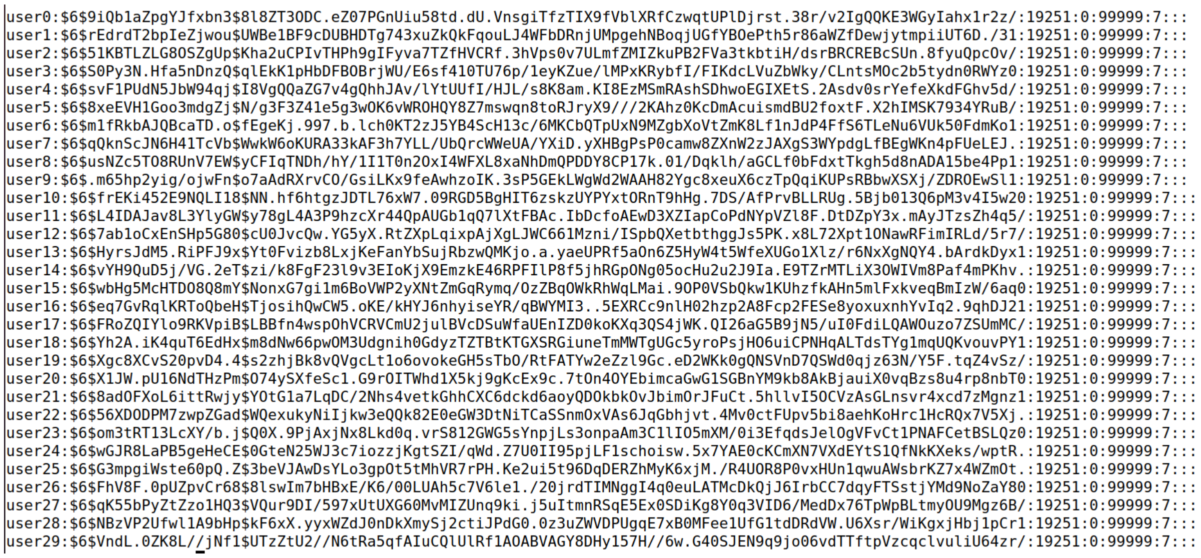


```
#script to add user0 to user29 in to the command line
def add_user(accounts):
    hashed_passwords = []
    for i in range(len(accounts)):
        username = accounts[i].getUsername()
        password = accounts[i].getPassword()

        encrypted_p = crypt.crypt(password)
        print(encrypted_p)

        hashed_passwords = encrypted_p

        subprocess.run(["sudo","useradd","-p",encrypted_p, username])

    return hashed_passwords

#add_user(accounts)


#to delete all the users if its needed

def delete_user(accounts):
    for i in range(len(accounts)):
        username = accounts[i].getUsername()
        password = accounts[i].getPassword()

        os.system("sudo deluser --remove-home "+username)


#delete_user(accounts)
```

In this method we actually add the user to the linux VM and to do that I need to iterate through every account in the list and get both the username and password. Once that is done, I need to use the crypt function on the current password to be able to encrypt the password because if that's missing, the shadow file will display the raw password. Once the password is encrypted then I use subproecess.run in order to be able to run the command that is used in the command line/terminal in order to create the users.

The last method was implemented only to remove all the users from the linux vm if a mistake was done.

b. **(5 points)** Display the contents of Password file and Shadow file. Which algorithm has been used to store the passwords?

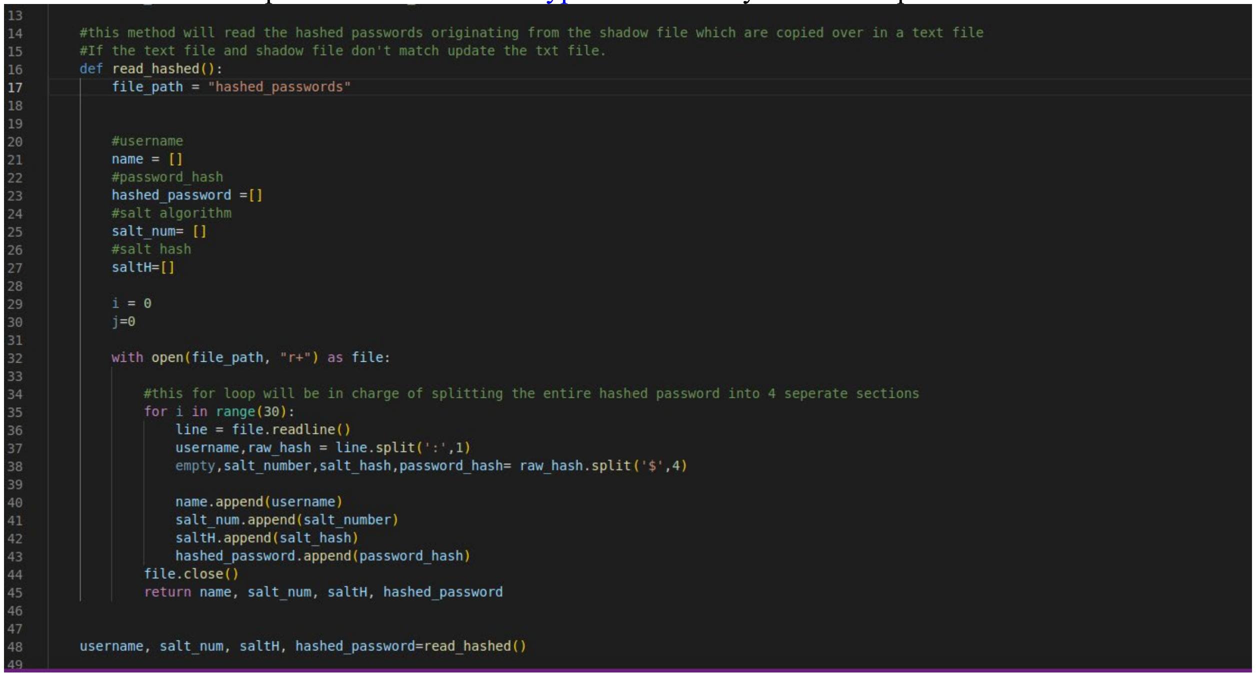


```
user0:$6$9iQb1aZpgYJfxbn3$8l8ZT3ODC.eZ07PGnUiu58td.dU.VnsgiTfzTIX9fVblXRfCzwqtUPlDjrst.38r/v2IgQQKE3WGyIahx1r2z/:19251:0:99999:7:::
user1:$6$rEdrdT2bpIeZjwou$UWBe1BF9cDUBHDTg743xuZkQkFqouLJ4WFbDRnjUMpgehNBoqjUGfYBOePth5r86aWZfDewjytmpiiUT6D./31:19251:0:99999:7:::
user2:$6$51KBTLZLG8OSZgUp$Kha2uCPIvTHPh9gIFyva7TZfHVCRf.3hVps0v7ULmfZMIZkuPB2FVa3tkbtiH/dsrBRCREBcSUn.8fyuQpcOv/:19251:0:99999:7:::
user3:$6$S0Py3N.Hfa5nDnzQ$qlEkK1pHbDFBOBrjWU/E6sf410TU76p/1eyKZue/lMPxKRybfI/FIKdcLVuZbWky/CLntsMOc2b5tydn0RWYz0:19251:0:99999:7:::
user4:$6$svF1PUdN5JbW94qj$I8VgQQaZG7v4gQhhJAv/lYtUUfI/HJL/s8K8am.KI8EzMSmRAshSDhwoEGIXEtS.2Asdv0srYefeXkdFGhv5d/:19251:0:99999:7:::
user5:$6$8xeEVH1Goo3mdgZj$N/g3F3Z41e5g3wOK6vWROHQY8Z7mswqn8toRJryX9///2KAhz0KcDmAcuismdBU2foxtF.X2hIMSK7934YRuB/:19251:0:99999:7:::
user6:$6$m1fRkbAJQBcaTD.o$fEgeKj.997.b.lch0KT2zJ5YB4ScH13c/6MKCbQTpUxN9MZgbXoVtZmK8Lf1nJdP4FfS6TLeNu6VUk50FdmKo1:19251:0:99999:7:::
user7:$6$qQknScJN6H41TcVb$WwkW6oKURA33kAF3h7YLL/UbQrcWWeUA/YXiD.yXHBgPsP0camw8ZXnW2zJAXgS3WYpdgLfBEgWKn4pFUeLEJ.:19251:0:99999:7:::
user8:$6$usNZc5TO8RUnV7EW$yCFIqTNDh/hY/1I1T0n2OxI4WFXL8xaNhDmQPDDY8CP17k.01/Dqklh/aGCLf0bFdxtTkgh5d8nADA15be4Pp1:19251:0:99999:7:::
user9:$6$.m65hp2yig/ojwFn$o7aAdRXrvCO/GsiLKx9feAwhzoIK.3sP5GEkLWgWd2WAAH82Ygc8xeuX6czTpQqiKUPsRBbwXSXj/ZDROEwSl1:19251:0:99999:7:::
user10:$6$frEKi452E9NQLI18$NN.hf6htgzJDTL76xW7.09RGD5BgHIT6zskzUYPYxtORnT9hHg.7DS/AfPrvBLLRUg.5Bjb013Q6pM3v4I5w20:19251:0:99999:7:::
user11:$6$L4IDAJav8L3YlyGW$y78gL4A3P9hzcXr44QpAUGb1qQ7lXtFBAc.IbDcfoAEwD3XZIapCoPdNYpVZl8F.DtDZpY3x.mAyJTzsZh4q5/:19251:0:99999:7:::
user12:$6$7ab1oCxEnSHp5G80$cU0JvcQw.YG5yX.RtZXpLqixpAjXgLJWC661Mzni/ISpbQXetbthggJs5PK.x8L72Xpt1ONawRFimIRLd/5r7/:19251:0:99999:7:::
user13:$6$HyrsJdM5.RiPFJ9x$Yt0Fvizb8LxjKeFanYbSujRbzwQMKjo.a.yaeUPRf5aOn6Z5HyW4t5WfeXUGo1Xlz/r6NxXgNQY4.bArdkDyx1:19251:0:99999:7:::
user14:$6$vYH9QuD5j/VG.2eT$zi/k8FgF23l9v3EIoKjX9EmzkE46RPFIlP8f5jhRGpONg05ocHu2u2J9Ia.E9TZrMTLiX3OWIVm8Paf4mPKhv.:19251:0:99999:7:::
user15:$6$wbHg5McHTDO8Q8mY$NonxG7gi1m6BoVWP2yXNtZmGqRymq/OzZBqOWkRhWqLMai.9OP0VSbQkw1KUhzfkAHn5mlFxkveqBmIzW/6aq0:19251:0:99999:7:::
user16:$6$eq7GvRqlKRToQbeH$TjosihQwCW5.oKE/kHYJ6nhyiseYR/qBWYMI3..5EXRCc9nlH02hzp2A8Fcp2FESe8yoxuxnhYvIq2.9qhDJ21:19251:0:99999:7:::
user17:$6$FRoZQIYlo9RKVpiB$LBBfn4wspOhVCRVCmU2julBVcDSuWfaUEnIZD0koKXq3QS4jWK.QI26aG5B9jN5/uI0FdiLQAWOuzo7ZSUmMC/:19251:0:99999:7:::
user18:$6$Yh2A.iK4quT6EdHx$m8dNw66pwOM3Udgnih0GdyzTZTBtKTGXSRGiuneTmMWTgUGc5yroPsjHO6uiCPNHqALTdsTYg1mqUQKvouvPY1:19251:0:99999:7:::
user19:$6$Xgc8XCvS20pvD4.4$s2zhjBk8vQVgcLt1o6ovokeGH5sTbO/RtFATYw2eZzl9Gc.eD2WKk0gQNSVnD7QSWd0qjz63N/Y5F.tqZ4vSz/:19251:0:99999:7:::
user20:$6$X1JW.pU16NdTHzPm$O74ySXfeSc1.G9rOITWhd1X5kj9gKcEx9c.7tOn4OYEbimcaGwG1SGBnYM9kb8AkBjauiX0vqBzs8u4rp8nbT0:19251:0:99999:7:::
user21:$6$8adOFXoL6ittRwjy$YOtG1a7LqDC/2Nhs4vetkGhhCXC6dckd6aoyQDOkbkOvJbimOrJFuCt.5hllvI5OCVzAsGLnsvr4xcd7zMgnz1:19251:0:99999:7:::
user22:$6$56XDODPM7zwpZGad$WQexukyNiIjkw3eQQk82E0eGW3DtNiTCaSSnmOxVAs6JqGbhjvt.4Mv0ctFUpv5bi8aehKoHrc1HcRQx7V5Xj.:19251:0:99999:7:::
user23:$6$om3tRT13LcXY/b.j$Q0X.9PjAxjNx8Lkd0q.vrS812GWG5sYnpjLs3onpaAm3C1lIO5mXM/0i3EfqdsJelOgVFvCt1PNAFCetBSLQz0:19251:0:99999:7:::
user24:$6$wGJR8LaPB5geHeCE$0GteN25WJ3c7iozzjKgtSZI/qWd.Z7U0II95pjLF1schoisw.5x7YAE0cKCmXN7VXdEYtS1QfNkKXeks/wptR.:19251:0:99999:7:::
user25:$6$G3mpgiWste60pQ.Z$3beVJAwDsYLo3gpOt5tMhVR7rPH.Ke2ui5t96DqDERZhMyK6xjM./R4UOR8P0vxHUn1qwuAWsbrKZ7x4WZmOt.:19251:0:99999:7:::
user26:$6$FhV8F.0pUZpvCr68$8lswIm7bHBxE/K6/00LUAh5c7V6le1./20jrdTIMNggI4q0euLATMcDkQjJ6IrbCC7dqyFTSstjYMd9NoZaY80:19251:0:99999:7:::
user27:$6$qK55bPyZtZzo1HQ3$VQur9DI/597xUtUXG60MvMIZUnq9ki.j5uItmnRSqE5Ex0SDiKg8Y0q3VID6/MedDx76TpWpBLtmyOU9Mgz6B/:19251:0:99999:7:::
user28:$6$NBzVP2Ufwl1A9bHp$kF6xX.yyxWZdJ0nDkXmySj2ctiJPdG0.0z3uZWVDPUgqE7xB0MFee1UfG1tdDRdVW.U6Xsr/WiKgxjHbj1pCr1:19251:0:99999:7:::
user29:$6$VndL.0ZK8L//jNf1$UTzZtU2//N6tRa5qfAIuCQlUlRf1AOABVAGY8DHy157H//6w.G40SJEN9q9jo06vdTTftpVzcqclvuliU64zr/:19251:0:99999:7:::
```

Alexis Navarro
88757719

The algorithm used to hash is “6” or which I learned later on that it uses SHA512 from the crypt.

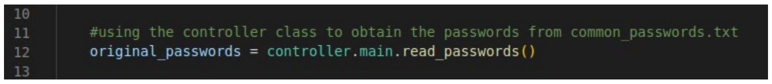
c. **(20 points)** Write a Python program to compute the necessary hash of the passwords (commonPasswdFile.txt) to compare them with the passwords stored in shadow file. The program should output all the username: password combinations after successful crack. Please list all the cracked combinations in the report. You can use the crypt function in Python to compute hash.



```python

    #this method will read the hashed passwords originating from the shadow file which are copied over in a text file
    #If the text file and shadow file don't match update the txt file.
    def read_hashed():
        file_path = "hashed_passwords"


        #username
        name = []
        #password_hash
        hashed_password =[]
        #salt algorithm
        salt_num= []
        #salt hash
        saltH=[]

        i = 0
        j=0

        with open(file_path, "r+") as file:

            #this for loop will be in charge of splitting the entire hashed password into 4 seperate sections
            for i in range(30):
                line = file.readline()
                username,raw_hash = line.split(':',1)
                empty,salt_number,salt_hash,password_hash= raw_hash.split('$',4)

                name.append(username)
                salt_num.append(salt_number)
                saltH.append(salt_hash)
                hashed_password.append(password_hash)
            file.close()
            return name, salt_num, saltH, hashed_password


    username, salt_num, saltH, hashed_password=read_hashed()

```

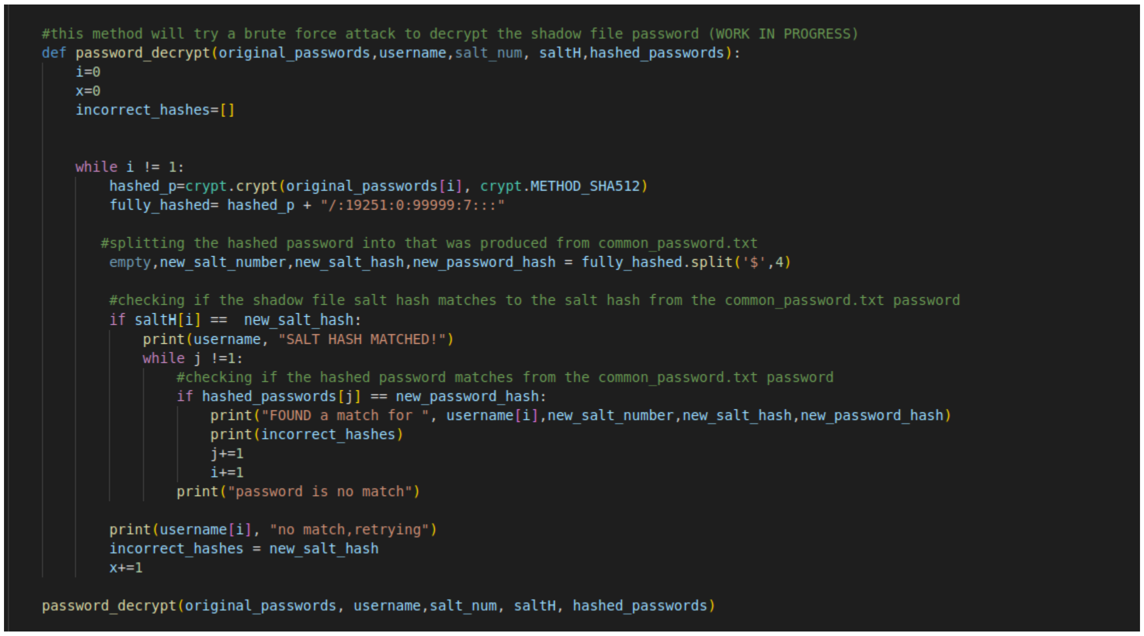
for this I created a new class named decrypt.py which will have the decryption functionality stored there. To begin I first made a new text file which I copied the contents of the user and the hashed password from the shadow file, just because reading the terminal seemed difficult for me to do at the time. Once that was set, I made the variable name which stores the usernames, the variable hashed_password which stores the hashed password, the salt_num which stores the algorithm used, then the saltH which is the section that follows after the algorithm used and before the hashed password is shown. To explain why I did this is because I assumed that if I split the sections of a password apart and generated the salt hash and the hashed passwords to compare later on, then I would be able to get a match much quicker.

Once I opened the file, I made sure to split the username from the hash then I split the hash every time we saw a “$” which indicates that it’s a different section of the hash. After doing so, I appended every piece that was split into the correct list. NOTE: the variable empty in the splitting holds an empty string which didn’t seem to have a use of keeping at the time.



```python

    #using the controller class to obtain the passwords from common_passwords.txt
    original_passwords = controller.main.read_passwords()

```

Alexis Navarro
88757719



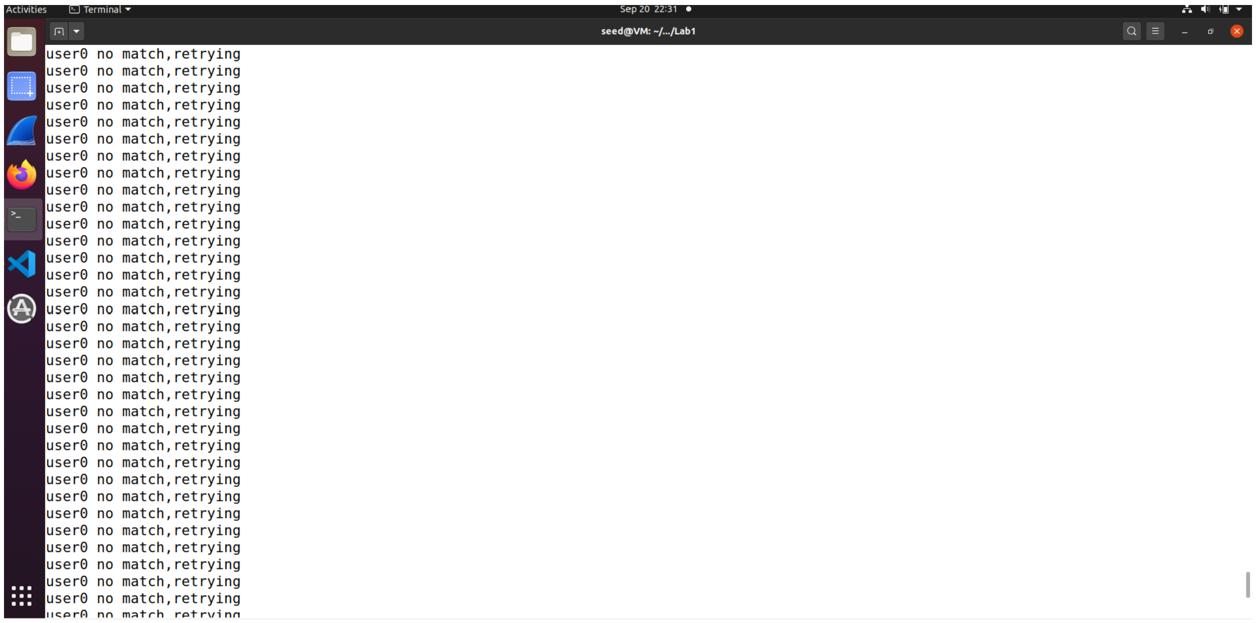
```python
#this method will try a brute force attack to decrypt the shadow file password (WORK IN PROGRESS)
def password_decrypt(original_passwords,username,salt_num, saltH,hashed_passwords):
    i=0
    x=0
    incorrect_hashes=[]


    while i != 1:
        hashed_p=crypt.crypt(original_passwords[i], crypt.METHOD_SHA512)
        fully_hashed= hashed_p + "/:19251:0:99999:7:::"

       #splitting the hashed password into that was produced from common_password.txt
        empty,new_salt_number,new_salt_hash,new_password_hash = fully_hashed.split('$',4)

        #checking if the shadow file salt hash matches to the salt hash from the common_password.txt password
        if saltH[i] ==  new_salt_hash:
            print(username, "SALT HASH MATCHED!")
            while j !=1:
                #checking if the hashed password matches from the common_password.txt password
                if hashed_passwords[j] == new_password_hash:
                    print("FOUND a match for ", username[i],new_salt_number,new_salt_hash,new_password_hash)
                    print(incorrect_hashes)
                    j+=1
                    i+=1
                print("password is no match")

        print(username[i], "no match,retrying")
        incorrect_hashes = new_salt_hash
        x+=1

password_decrypt(original_passwords, username,salt_num, saltH, hashed_passwords)
```

Now to this the decrypting I passed the lists that had the separated sections of the users hashed passwords and the original passwords from the text files without the hash. In the code I only planned to decrypt the first users password instead of all 30 since it would take an extremely long time to brute force the decryption. To begin I hashed the raw password using the algorithm that crypt used in the previous class controller.py, once I encrypted the password, I added the section at the very end of the shadow file that I accidentally forgot to remove when splitting the hashed password.

Soon after I repeated the method of splitting the new password in to parts just to be able to speed up comparisons and matching time. After that was done, I began to check the saltH(hashed salt) from the shadow file with the new hash value that we generated here, if we found it then we would print the user name with a statement saying that there was a match. Now that we passed the first section, we would compare the hashed_passwords(the hashed password from the shadow file) with the new password hash generated by the commonPasswords.txt file. If there was a match then we print that we found a match along with the username and entire hash contents of it. If we never got a match in any of the two sections, then we would store the hashes in an incorrect hash list and retry.

NOTE: I didn't have much time to implement the incorrect hash list, since I believed I would increase the time complexity of the system if I would check a list of incorrect hashes with the current one to see if there were no duplicates.

Alexis Navarro
88757719



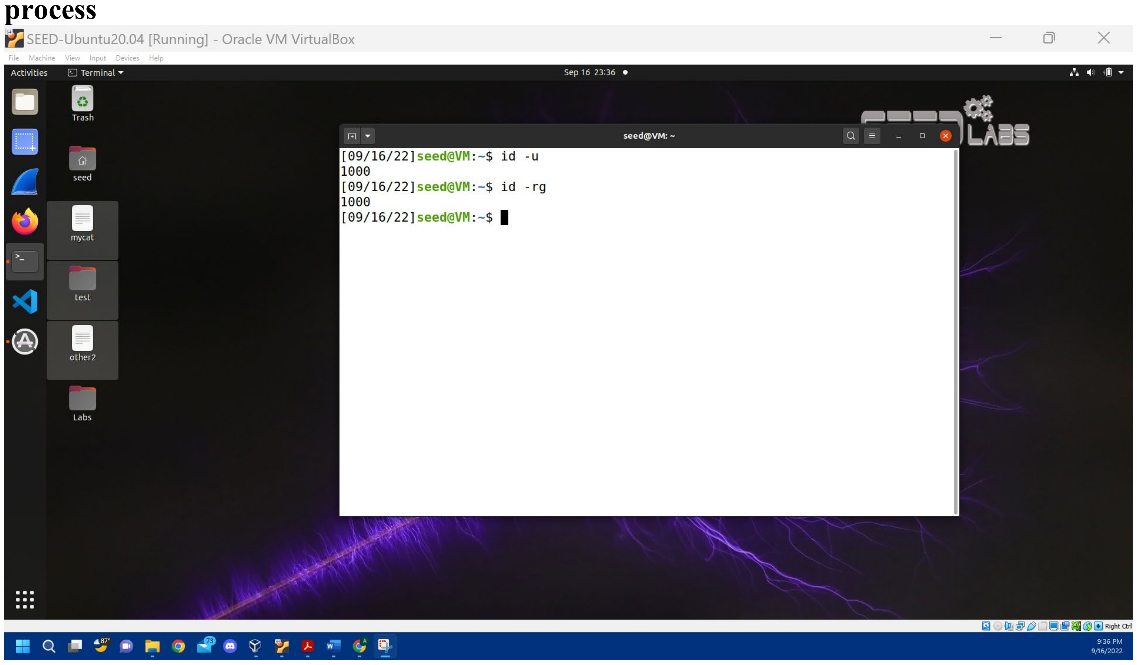
```
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
user0 no match,retrying
```

In the end I would never find a match since I could only assume that eventually I will find a match but at the worst possible scenario which it would be after a couple of hours or days or until the vm breaks from the amount of tries. I believed that EVENTUALLY I'll obtain the password since this code is brute forcing its way to decrypt the password.

Question 3: Please answer the following questions.

a. **(5 points)** What is real User ID (UID) and Effective User ID (EUID)? Please provide the screenshot of UID and EUID for seed user?

   a. **A real user id is the real owner of a process and effective user id is the privilege of a process**

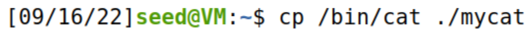
```
[09/16/22]seed@VM:~$ id -u
1000
[09/16/22]seed@VM:~$ id -rg
1000
[09/16/22]seed@VM:~$ 
```

   b.

   c. Explanation: the first line is to be able to check what the effective user id is currently, the next line in the terminal is to check the real Id.

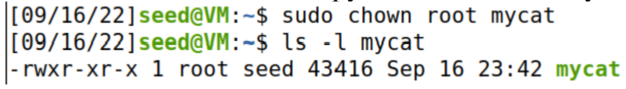
Alexis Navarro
88757719

b. **(5 points)** Please provide a step-by-step demonstration of how to Set-UID programs can be used to display contents of shadow file for seed user.

a. 

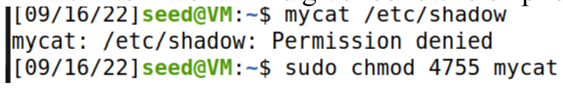
```
[09/16/22]seed@VM:~$ cp /bin/cat ./mycat
```

i. We first need to copy the cat file in the system to mycat

b. 

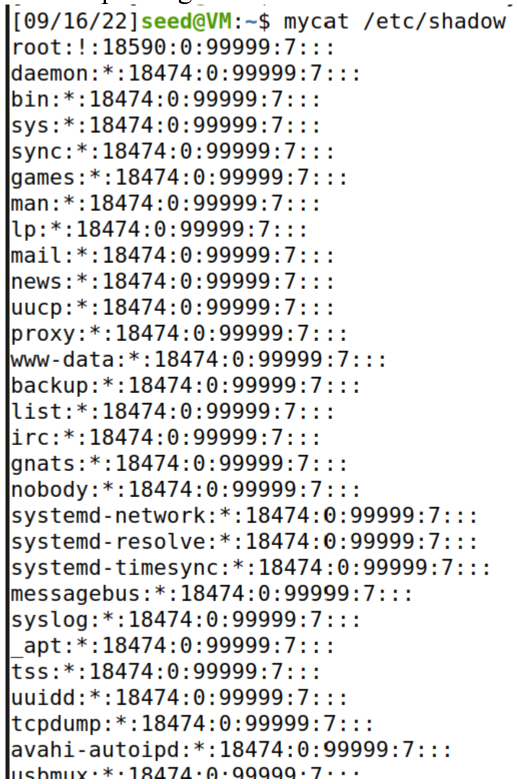
```
[09/16/22]seed@VM:~$ sudo chown root mycat
[09/16/22]seed@VM:~$ ls -l mycat
-rwxr-xr-x 1 root seed 43416 Sep 16 23:42 mycat
```

i. Next we want to give root ownership to mycat

c. 

```
[09/16/22]seed@VM:~$ mycat /etc/shadow
mycat: /etc/shadow: Permission denied
[09/16/22]seed@VM:~$ sudo chmod 4755 mycat
```

i. Once we're denied access to the shadow file, we want to have mycat get elevated privileges to be able to access the shadow file as a root user

d. 

```
[09/16/22]seed@VM:~$ mycat /etc/shadow
root:!:18590:0:99999:7:::
daemon:*:18474:0:99999:7:::
bin:*:18474:0:99999:7:::
sys:*:18474:0:99999:7:::
sync:*:18474:0:99999:7:::
games:*:18474:0:99999:7:::
man:*:18474:0:99999:7:::
lp:*:18474:0:99999:7:::
mail:*:18474:0:99999:7:::
news:*:18474:0:99999:7:::
uucp:*:18474:0:99999:7:::
proxy:*:18474:0:99999:7:::
www-data:*:18474:0:99999:7:::
backup:*:18474:0:99999:7:::
list:*:18474:0:99999:7:::
irc:*:18474:0:99999:7:::
gnats:*:18474:0:99999:7:::
nobody:*:18474:0:99999:7:::
systemd-network:*:18474:0:99999:7:::
systemd-resolve:*:18474:0:99999:7:::
systemd-timesync:*:18474:0:99999:7:::
messagebus:*:18474:0:99999:7:::
syslog:*:18474:0:99999:7:::
_apt:*:18474:0:99999:7:::
tss:*:18474:0:99999:7:::
uuidd:*:18474:0:99999:7:::
tcpdump:*:18474:0:99999:7:::
avahi-autoipd:*:18474:0:99999:7:::
usbmux:*:18474:0:99999:7:::
```

i. Once we use mycat and access the shadow file, we can see the information that should be listed exclusively to the root user, in here by changing our effective user ID to root.

c. **(10 points)** How does Set-UID programs affect EUID? Please explain with an example.

a. So Set-UID programs give us the opportunity to allow non-root users to be able to access root exclusive programs once set-uid is applied to those programs.

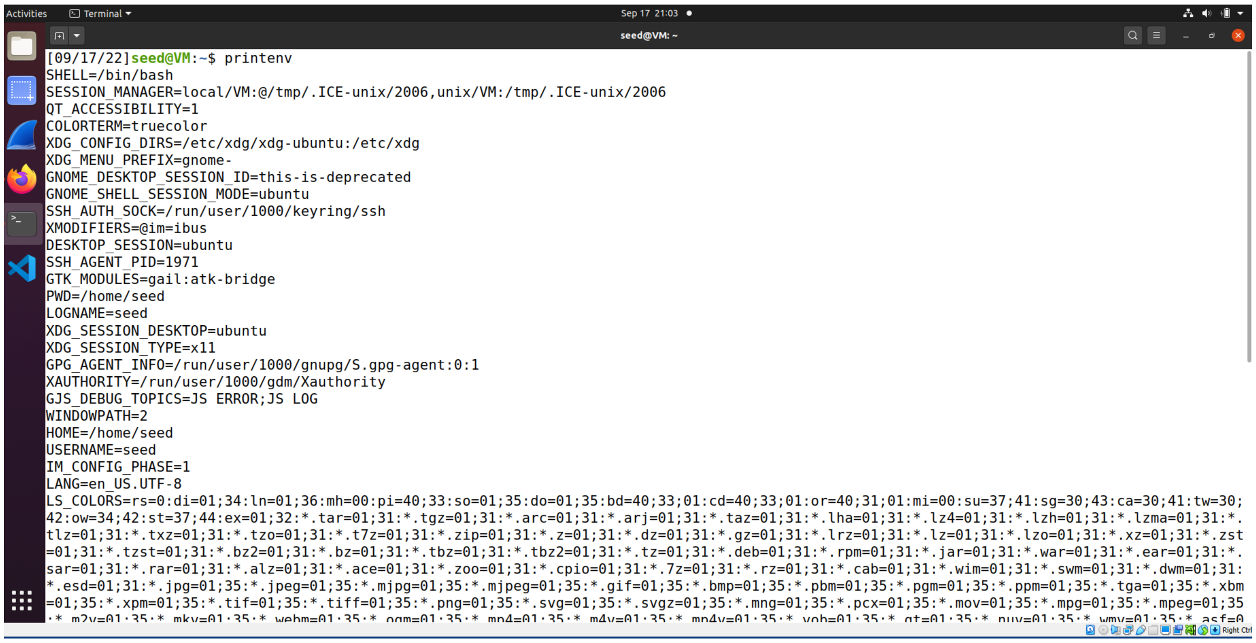
b. To start Effective User ID (EUID) is the same number as Real User ID(UID) in which this case let's say EUID = 1000 and UID = 1000

c. Without using set-uid in programs, EUID and UID should stay the same value, but if we use want to use a program that only allows the root to access and is represented by "-rwxr" at the start then we would use the SET-UID to gain access to that program.

Alexis Navarro
88757719

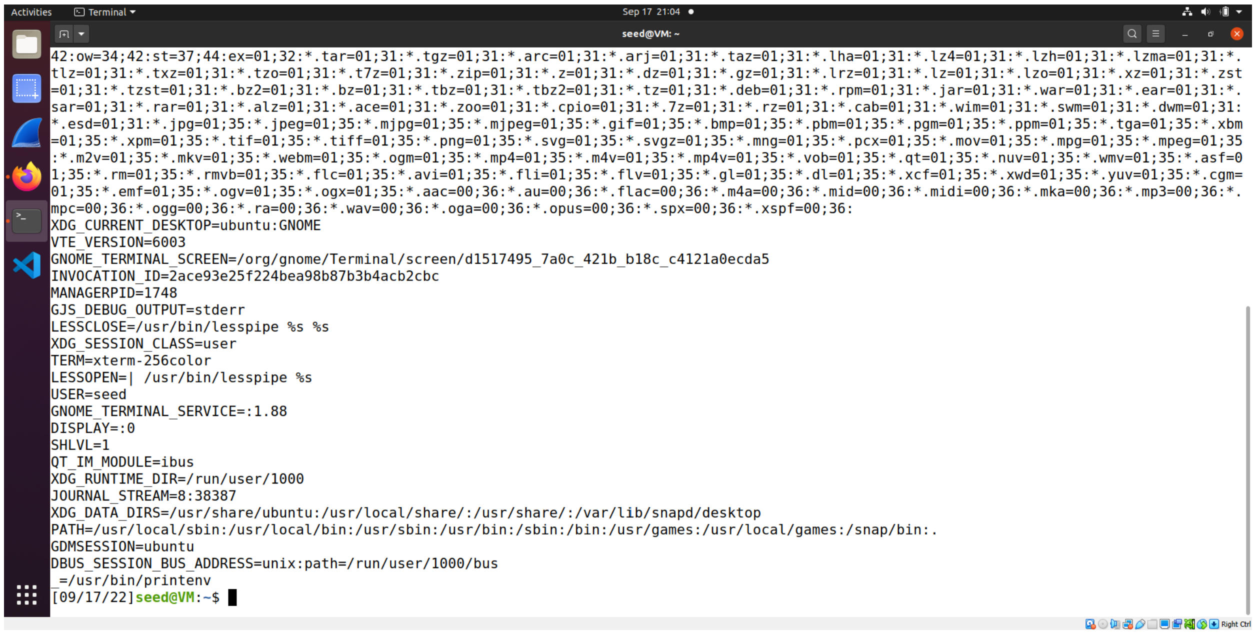
d. When we use Set-UID, we can see that the programs permissions changed to "-rwsr" the S meaning that this is an elevated program now. However doing this, the Effective User ID will change to what the root ID is which will be EUID = 0.
e. With the effective user Id at 0, our real id will remain at 1000 since that is the original id that the user is under on.

(20 Points) Task 1: Manipulating Environment Variables
In this task, we study the commands that can be used to set and unset environment variables. We are using Bash in the seed account. The default shell that a user use is set in the /etc/passwd file (the last field of each entry). You can change this to another shell program using the command chsh (please do not do it for this lab). Please do the following tasks:
• • Use printenv or env command to print out the environment variables. If you are interested in some environment variables, such as PWD, you can use "printenv PWD" or "env |grep PWD".



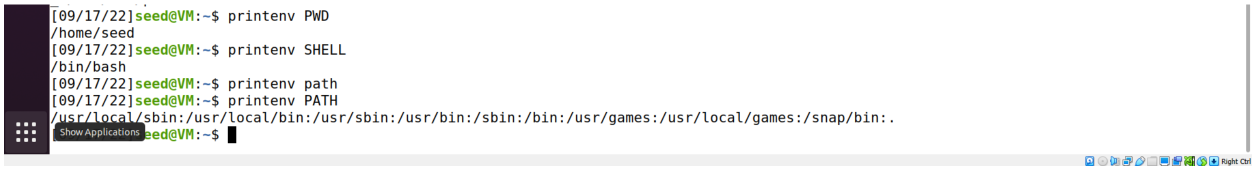
```
[09/17/22]seed@VM:~$ printenv
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/2006,unix/VM:/tmp/.ICE-unix/2006
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=1971
GTK_MODULES=gail:atk-bridge
PWD=/home/seed
LOGNAME=seed
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
XAUTHORITY=/run/user/1000/gdm/Xauthority
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
WINDOWPATH=2
HOME=/home/seed
USERNAME=seed
IM_CONFIG_PHASE=1
LANG=en_US.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;
42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.
tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst
=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.
sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:
*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm
=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35
:*.m2v=01:35:*.mkv=01:35:*.webm=01:35:*.ogm=01:35:*.mp4=01:35:*.m4v=01:35:*.mp4v=01:35:*.vob=01:35:*.qt=01:35:*.nuv=01:35:*.wmv=01:35:*.asf=0
```

Alexis Navarro
88757719



```
42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.
tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst
=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.
sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:
*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm
=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35
:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=0
1;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=
01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.
mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
XDG_CURRENT_DESKTOP=ubuntu:GNOME
VTE_VERSION=6003
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/d1517495_7a0c_421b_b18c_c4121a0ecda5
INVOCATION_ID=2ace93e25f224bea98b87b3b4acb2cbc
MANAGERPID=1748
GJS_DEBUG_OUTPUT=stderr
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
GNOME_TERMINAL_SERVICE=:1.88
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=8:38387
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:.
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
_=/usr/bin/printenv
[09/17/22]seed@VM:~$ 
```

This last one I used printenv to be able to check specific environment variables and to see if it was case sensitive when calling an environment variable which turned out to be case sensitive.

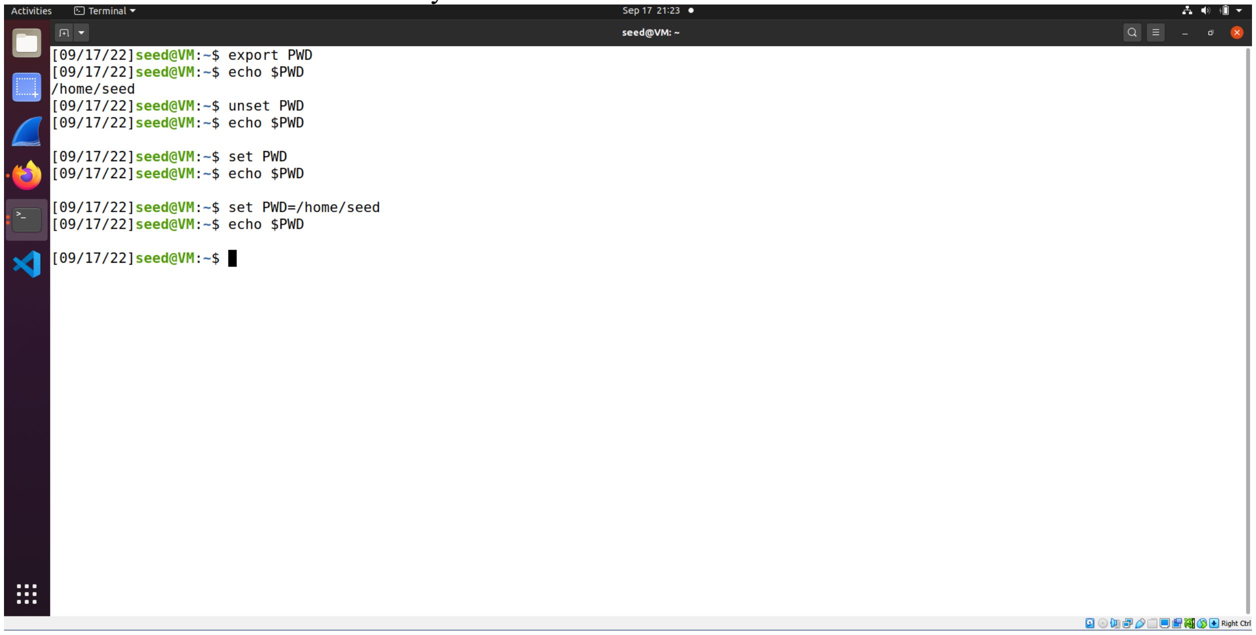


```
[09/17/22]seed@VM:~$ printenv PWD
/home/seed
[09/17/22]seed@VM:~$ printenv SHELL
/bin/bash
[09/17/22]seed@VM:~$ printenv path
[09/17/22]seed@VM:~$ printenv PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:.
          eed@VM:~$ 
```

Alexis Navarro
88757719

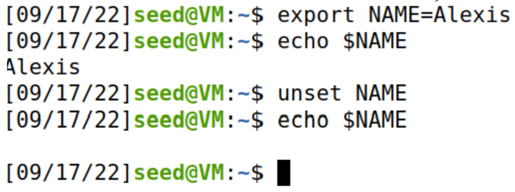
• Use export and unset to set or unset environment variables. It should be noted that these two commands are not separate programs; they are two of the Bash's internal commands (you will not be able to find them outside of Bash).

In this part of the terminal I am trying to use the "export" and "unset" commands when using an actual environment variable. In the first line, I start off with exporting PWD and then using echo to see the value of a variable of $PWD now which prints out what was listed in the environment variables which was "/home/seed". After I used the unset command on PWD and then used echo and could see that the value inside PWD was gone now. I even tried to set PWD to see if I can get the value back that was originally there but even with two different ways it seemed like I couldn't use set on this environment variable.



```
[09/17/22]seed@VM:~$ export PWD
[09/17/22]seed@VM:~$ echo $PWD
/home/seed
[09/17/22]seed@VM:~$ unset PWD
[09/17/22]seed@VM:~$ echo $PWD

[09/17/22]seed@VM:~$ set PWD
[09/17/22]seed@VM:~$ echo $PWD

[09/17/22]seed@VM:~$ set PWD=/home/seed
[09/17/22]seed@VM:~$ echo $PWD

[09/17/22]seed@VM:~$ 
```

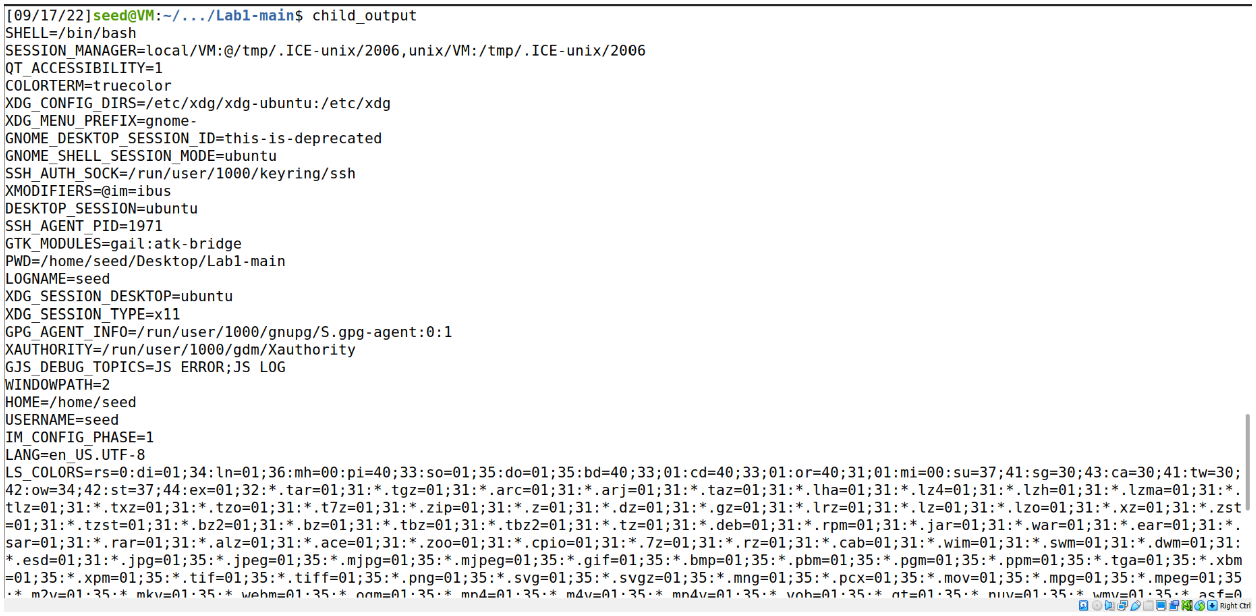
In this part I am assigning a variable name to see if it accepts "export" and "unset". For here I set a variable as "NAME" which has a value of "Alexis" and when I use export it appears to be assigning the value "Alexis" to NAME; I then use echo to make sure the value is still there. After I try to unset the variable NAME and once its echoed, I can see that the variable NAME has no value assigned.



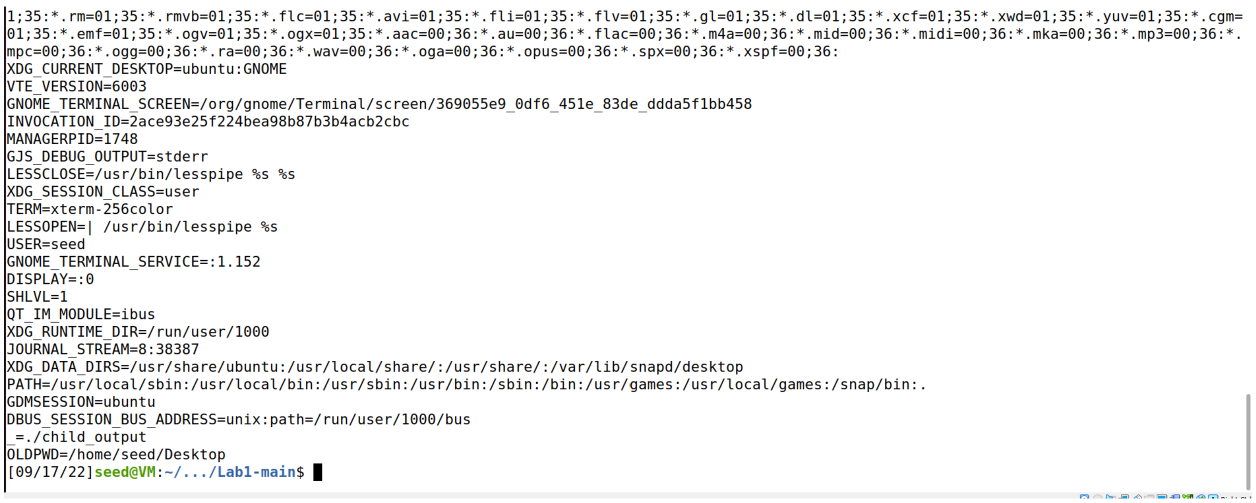
```
[09/17/22]seed@VM:~$ export NAME=Alexis
[09/17/22]seed@VM:~$ echo $NAME
Alexis
[09/17/22]seed@VM:~$ unset NAME
[09/17/22]seed@VM:~$ echo $NAME

[09/17/22]seed@VM:~$ 
```

Alexis Navarro
88757719

(30 Points) Task 2: Passing Environment Variables from Parent Process to Child Process
In this task, we study how a child process gets its environment variables from its parent. In Unix, fork () creates a new process by duplicating the calling process. The new process, referred to as the child, is an exact duplicate of the calling process, referred to as the parent; however, several things are not inherited by the child (please see the manual of fork () by typing the following command: man fork). In this task, we would like to know whether the parent's environment variables are inherited by the child process or not.
a. Please compile and run "myprintenv.c" and describe your observation. Because the output contains many strings, you should save the output into a file, such as using a.out > child (if a.out is your executable file name).



```
[09/17/22]seed@VM:~/.../Lab1-main$ child_output
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/2006,unix/VM:/tmp/.ICE-unix/2006
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=1971
GTK_MODULES=gail:atk-bridge
PWD=/home/seed/Desktop/Lab1-main
LOGNAME=seed
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
XAUTHORITY=/run/user/1000/gdm/Xauthority
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
WINDOWPATH=2
HOME=/home/seed
USERNAME=seed
IM_CONFIG_PHASE=1
LANG=en_US.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;
42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.
tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst
=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.
sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:
*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm
=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35
:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=0
```

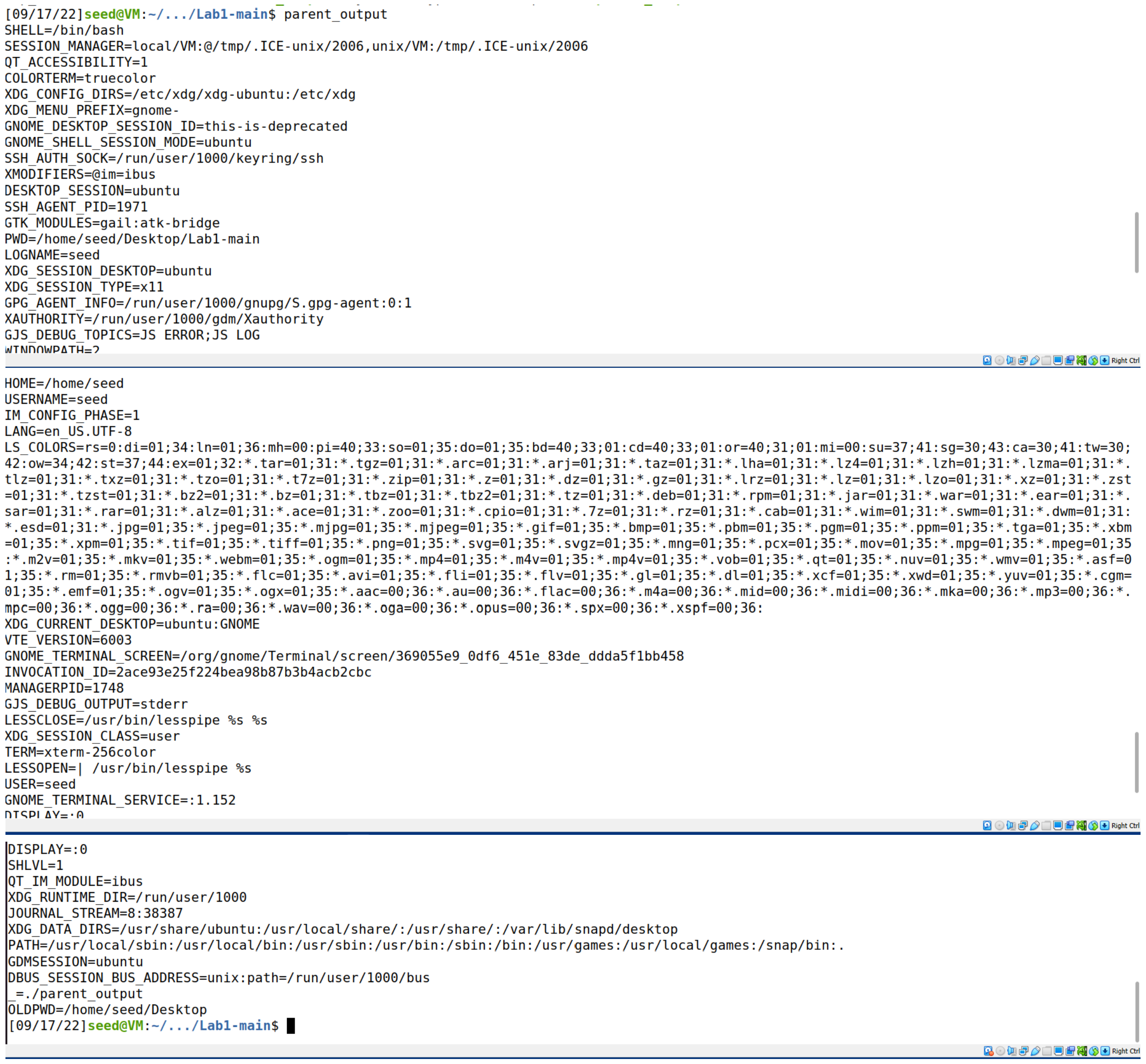


```
1;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=
01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.
mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
XDG_CURRENT_DESKTOP=ubuntu:GNOME
VTE_VERSION=6003
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/369055e9_0df6_451e_83de_ddda5f1bb458
INVOCATION_ID=2ace93e25f224bea98b87b3b4acb2cbc
MANAGERPID=1748
GJS_DEBUG_OUTPUT=stderr
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
GNOME_TERMINAL_SERVICE=:1.152
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=8:38387
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:.
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
_=./child_output
OLDPWD=/home/seed/Desktop
[09/17/22]seed@VM:~/.../Lab1-main$ 
```

When looking at the output, I can see that the output file contains the environment variables, some obvious ones being the PATH where the output is stored. But I did notice that in the very last part we have a DBUS_SESSION_BUS_ADDRESS which executes our child output which I can assume its unique to the executable.

Alexis Navarro
88757719

b. Now comment out the printenv() statement in the child process case and uncomment the printenv() statement in the parent process case. Compile and run the code again and describe your observation. Save the output in another file.



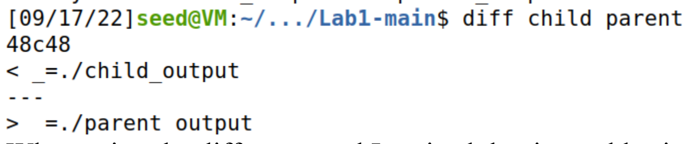
```
[09/17/22]seed@VM:~/.../Lab1-main$ parent_output
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/2006,unix/VM:/tmp/.ICE-unix/2006
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=1971
GTK_MODULES=gail:atk-bridge
PWD=/home/seed/Desktop/Lab1-main
LOGNAME=seed
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
XAUTHORITY=/run/user/1000/gdm/Xauthority
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
WINDOWPATH=2
```

```
HOME=/home/seed
USERNAME=seed
IM_CONFIG_PHASE=1
LANG=en_US.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;
42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.
tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst
=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.
sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:
*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm
=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35
:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=0
1;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=
01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.
mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
XDG_CURRENT_DESKTOP=ubuntu:GNOME
VTE_VERSION=6003
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/369055e9_0df6_451e_83de_ddda5f1bb458
INVOCATION_ID=2ace93e25f224bea98b87b3b4acb2cbc
MANAGERPID=1748
GJS_DEBUG_OUTPUT=stderr
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
GNOME_TERMINAL_SERVICE=:1.152
DISPLAY=:0
```

```
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=8:38387
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:.
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
_=./parent_output
OLDPWD=/home/seed/Desktop
[09/17/22]seed@VM:~/.../Lab1-main$
```

in the output for the parent the entire output looks almost the same to me as I went through it line by line, the only difference that I noticed was the DBUS_SESSION_BUS_ADDRESS which has the name of this current file that we ran.

c. Compare the difference of these two files using the diff command. Please draw your conclusion.



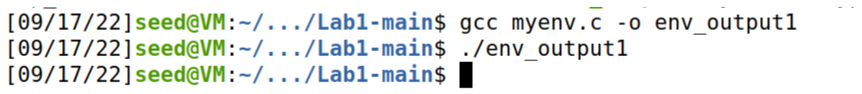
```
[09/17/22]seed@VM:~/.../Lab1-main$ diff child parent
48c48
< _=./child_output
---
> _=./parent_output
```

When using the diff command I noticed that it would print the items that were different from both the files in which it was the DBUS_SESSION_BUS_ADDRESS that was different since these are sent to two different output executable files.

Alexis Navarro
88757719

(30 Points) Task 3: Environment Variables, execve() and system()
In this task, we study how environment variables are affected when a new program is executed via execve(). The function execve() calls a system call to load a new command and execute it; this function never returns. No new process is created; instead, the calling process's text, data, bss, and stack are overwritten by that of the program loaded. Essentially, execve() runs the new program inside the calling process. We are interested in what happens to the environment variables; are they automatically inherited by the new program?

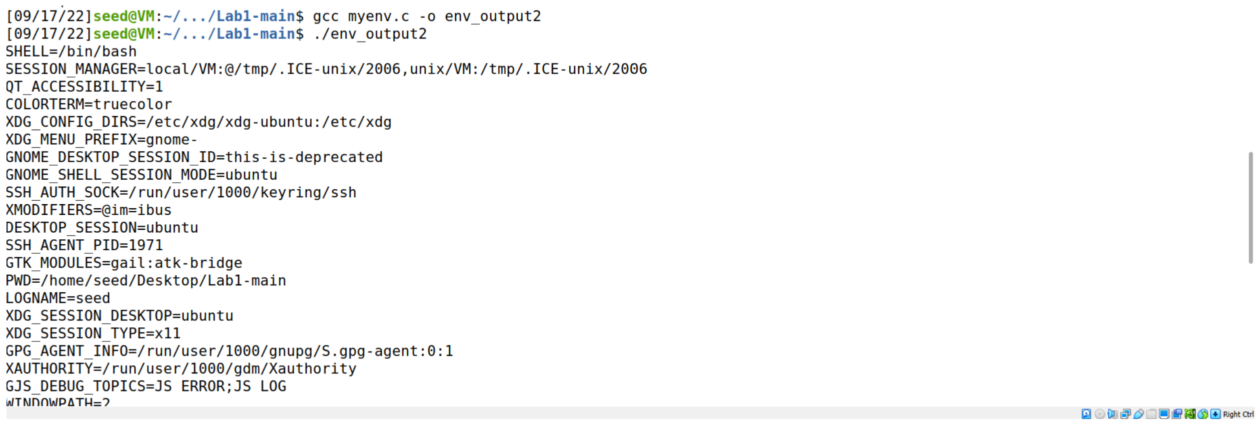
a) Please compile and run "myenv.c" and describe your observation. This program simply executes a program called /usr/bin/env, which prints out the environment variables of the current process.



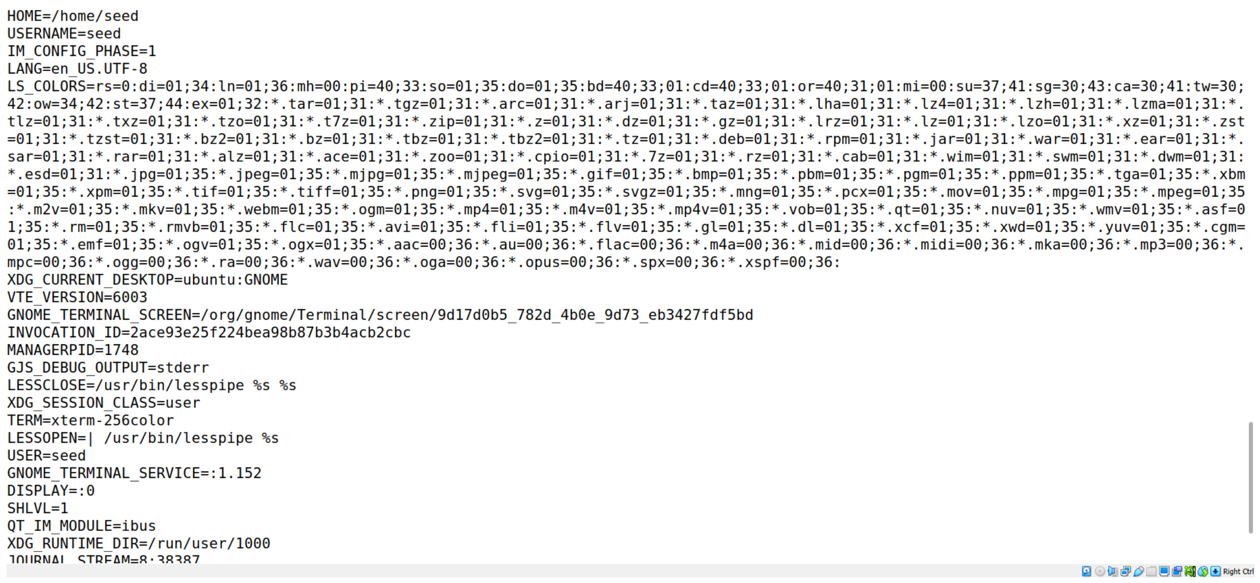
```
[09/17/22]seed@VM:~/.../Lab1-main$ gcc myenv.c -o env_output1
[09/17/22]seed@VM:~/.../Lab1-main$ ./env_output1
[09/17/22]seed@VM:~/.../Lab1-main$ 
```

for this output thereis nothing printed and I had to check the code to see what was the cause of it and it was " execve("/usr/bin/env", argv, NULL);" this line of code where NULL is preventing the printing of the environment variables.

b) Change the invocation of execve() in the code to the following line and describe your observation. execve("/usr/bin/env", argv, environ)

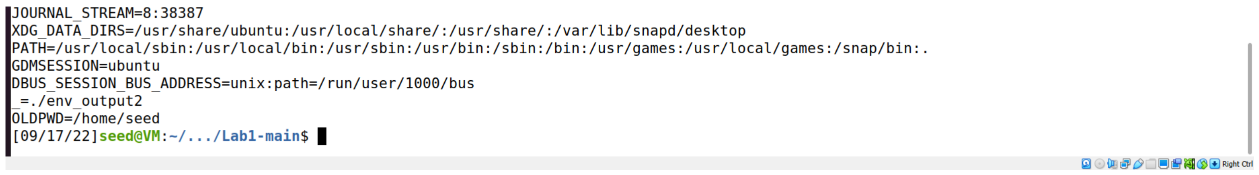


```
[09/17/22]seed@VM:~/.../Lab1-main$ gcc myenv.c -o env_output2
[09/17/22]seed@VM:~/.../Lab1-main$ ./env_output2
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/2006,unix/VM:/tmp/.ICE-unix/2006
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=1971
GTK_MODULES=gail:atk-bridge
PWD=/home/seed/Desktop/Lab1-main
LOGNAME=seed
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
XAUTHORITY=/run/user/1000/gdm/Xauthority
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
WINDOWPATH=2
```



```
HOME=/home/seed
USERNAME=seed
IM_CONFIG_PHASE=1
LANG=en_US.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;
42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.
tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst
=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.
sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:
*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm
=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35
:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=0
1;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=
01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.
mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
XDG_CURRENT_DESKTOP=ubuntu:GNOME
VTE_VERSION=6003
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/9d17d0b5_782d_4b0e_9d73_eb3427fdf5bd
INVOCATION_ID=2ace93e25f224bea98b87b3b4acb2cbc
MANAGERPID=1748
GJS_DEBUG_OUTPUT=stderr
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
GNOME_TERMINAL_SERVICE=:1.152
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=8:38387
```

Alexis Navarro
88757719



```
JOURNAL_STREAM=8:38387
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:.
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
_=./env_output2
OLDPWD=/home/seed
[09/17/22]seed@VM:~/.../Lab1-main$
```

Now that we changed the code, we can see that we print all the environment variables. The reason why this is happening is because before we hat the NULL value in the line of code preventing us to do anything such as printing, but once we placed the variable environ there we can see that we are able to print the environment variables since environ gives us access to the environment variables and its what we need for the 3rd parameter to be in execve() to print the environment variables.

c) Please draw your conclusion regarding how the new program gets its environment variables.

In the method that it gets its environment variables, I believe its able to do that since execve calls a system call to load a command and excutes it, but specifically execve is in charge of running it. Aside from that execve needs needs 3 parameters to be able to obtain the environment variables which in the first parameter you place the path, then the 2nd parameter needs to be argv since it needs an array of argument strings to be passed to the new program but needs to be terminated by a null pointer which in this case it be the last element in the array as NULL, and finally we need environ to be able to print the environment variables.

d) How would you use system() function to do the above task? Please explain the difference between system() and execve() system calls.

if I were to use system() I would specifically use os.system in order to be able to access the functions that the terminal has. Then in the parameters of using os.system() I would pass the variables that execve() contains such as the path, the arguments, and the environ variable.

The difference between these two is that system directly invokes a shell to be able to parse the string to accomplish the commands with in the terminal. I believe that execve doesn't interpret the arguments as a shell command, but it does allow us to set the environment to what we specify which makes it safer to pass commands to the shell to prevent any other command being called other than the thing we want.