# SetUID Lab

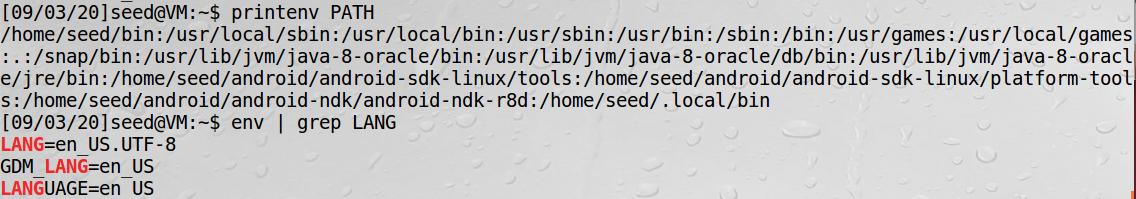
57117217 戚吴祺

## Task 1: Manipulating Environment Variables



The default shell in the seed account is bash. (/bin/bash)

Use printenv or env command to print out the environment variables.

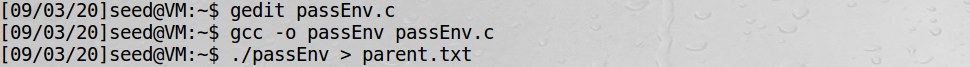


Use export and unset to set or unset environment variables.



## Task 2: Passing Environment Variables from Parent Process to Child Process

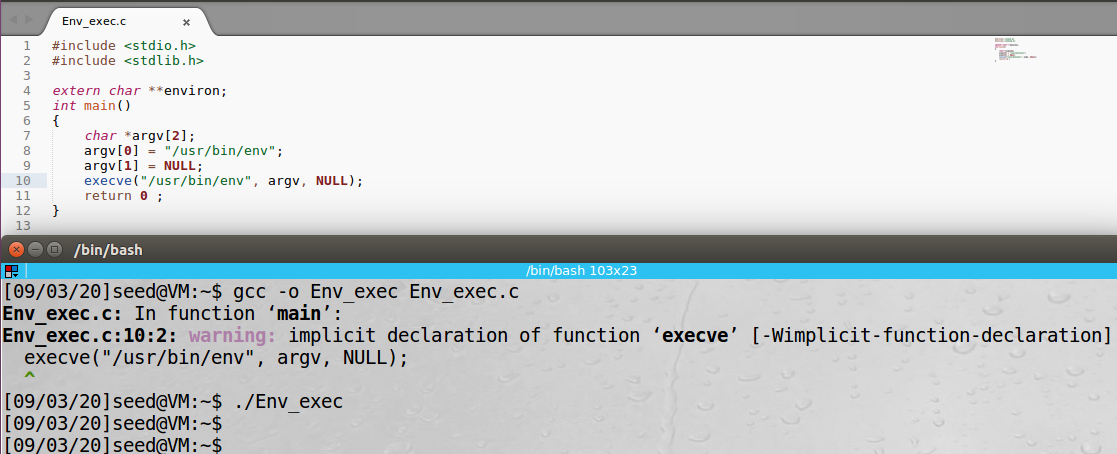




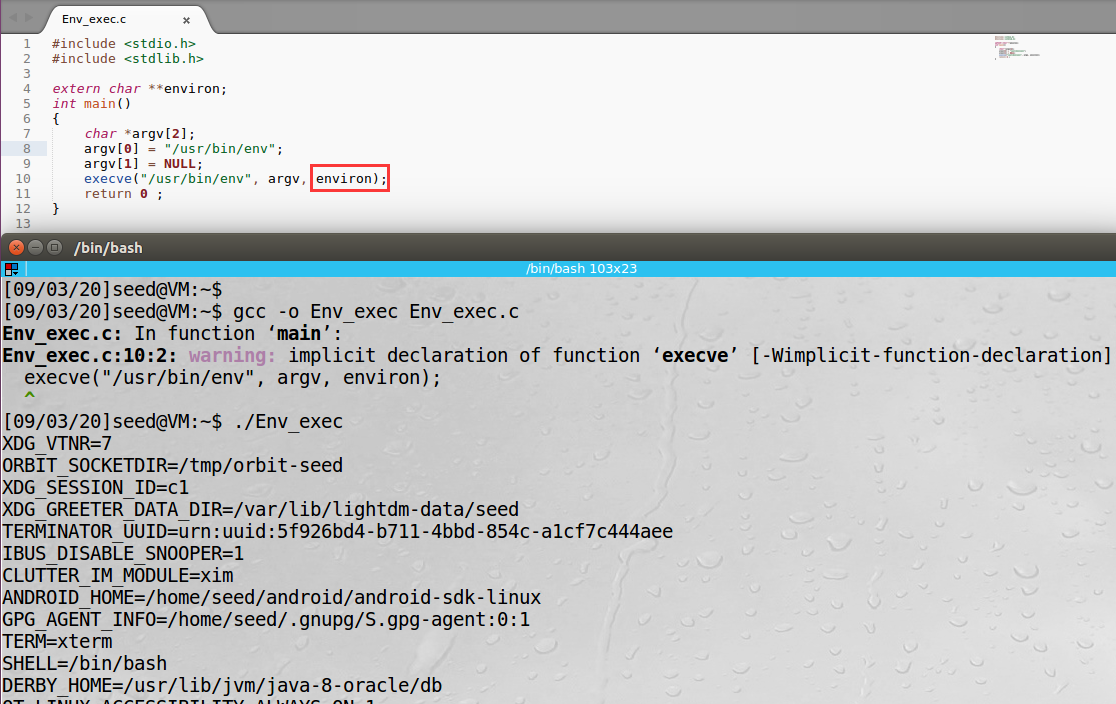
Conclusion: In this task, I know that the parent’s environment variables are inherited by the child process. Additionally, by manning fork, I understand that the child process inherits the process's qualifications, environment, stack, and memory stub directories from the parent process. However, the child process does not inherit some of the features of the parent process, such as the parent process number, file descriptor, system time in the TMS structure, resource usage, etc.

## Task 3: Environment Variables and execve()

*int execve(const char \* filename，char \* const argv[]，char \* const envp[])*



The environment variables of the current process were not printed out when using ‘execve("/usr/bin/env", argv, NULL)’. It’s NULL.



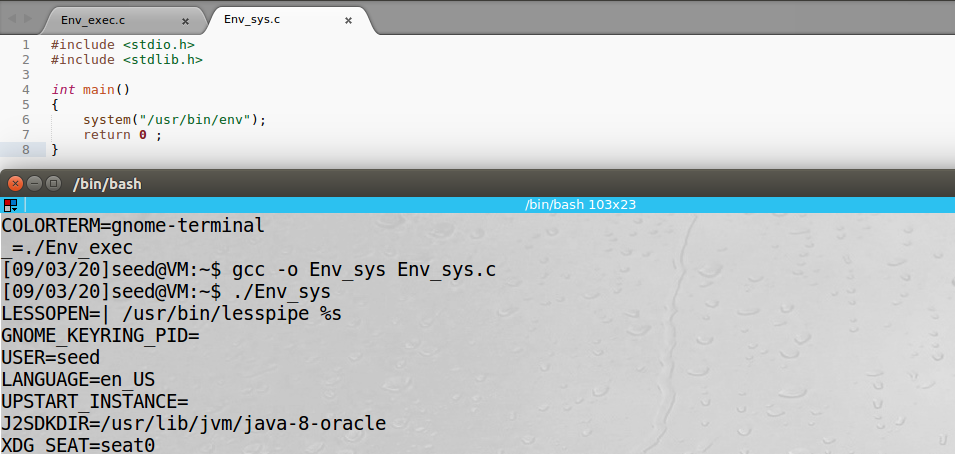
The environment variables of the current process were printed out when using ‘execve("/usr/bin/env", argv, environ)’.

Conclusion: The new program gets its environment variables by the parameter ‘env’ passed to the new program rather than inheriting from the old program.

## Task 4: Environment Variables and system()

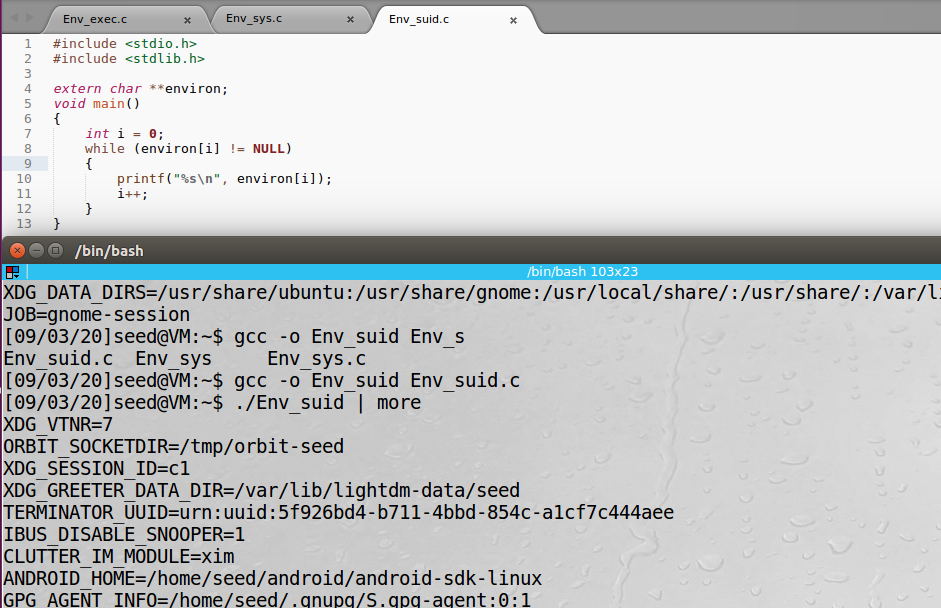
*Function system()*



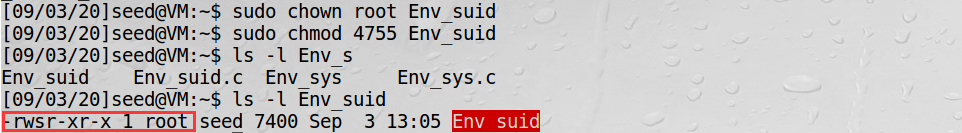


The environment variables of the calling process is passed to the new program /usr/bin/env.

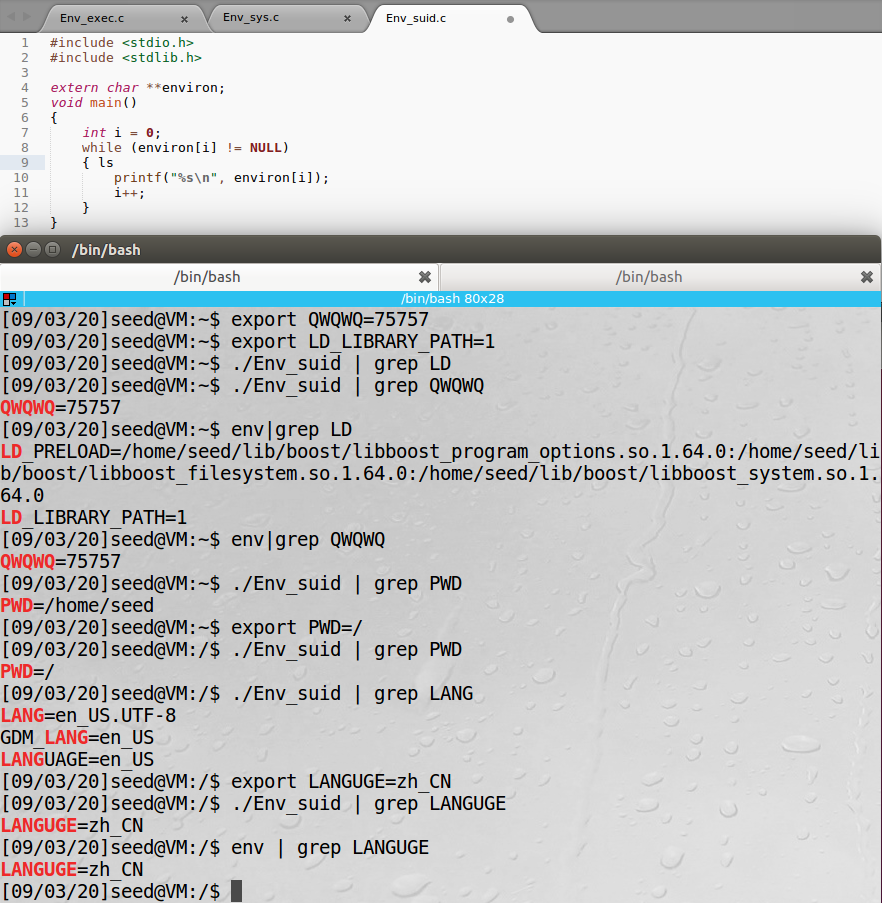
## Task 5: Environment Variable and Set-UID Programs



The program prints out all the environment variables in the current process.

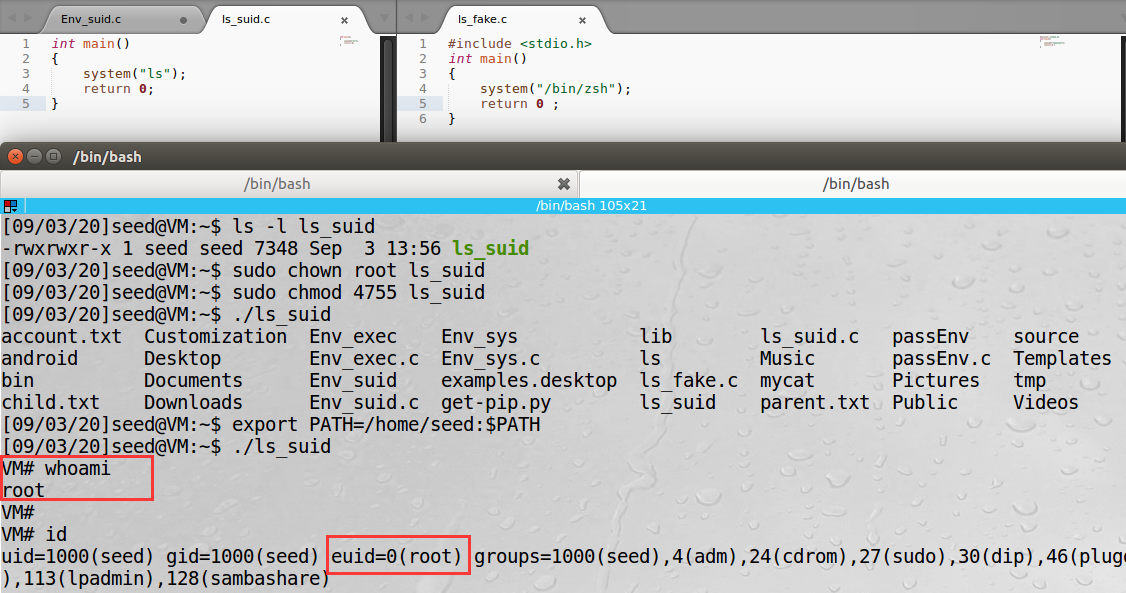


Change its ownership to root, and make it a Set-UID program.



These environment variables set in the shell process can only be used in this shell. After modifying the system environment variables, it will success when a suid program using changed variables. All the environment variables i set in the shell process (parent) get into the Set-UID child process.

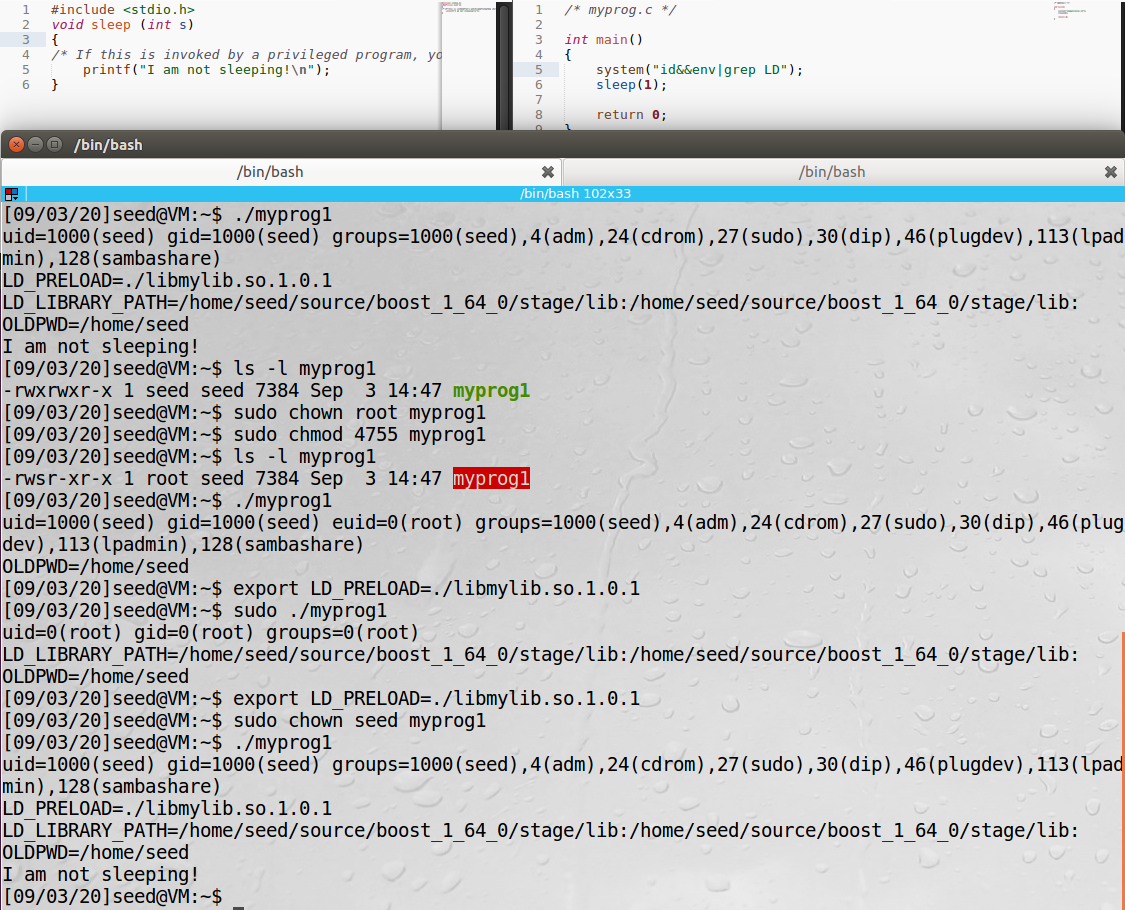
## Task 6: The PATH Environment Variable and Set-UID Programs



My code ran with the root privilege. This is because the actual behavior of the shell program can be affected by environment variables, such as ‘PATH=/home/seed:$PATH’. The program execute the fake ls in dir /home/seed/ls first.

## Task 7: The LD PRELOAD Environment Variable and Set-UID Programs

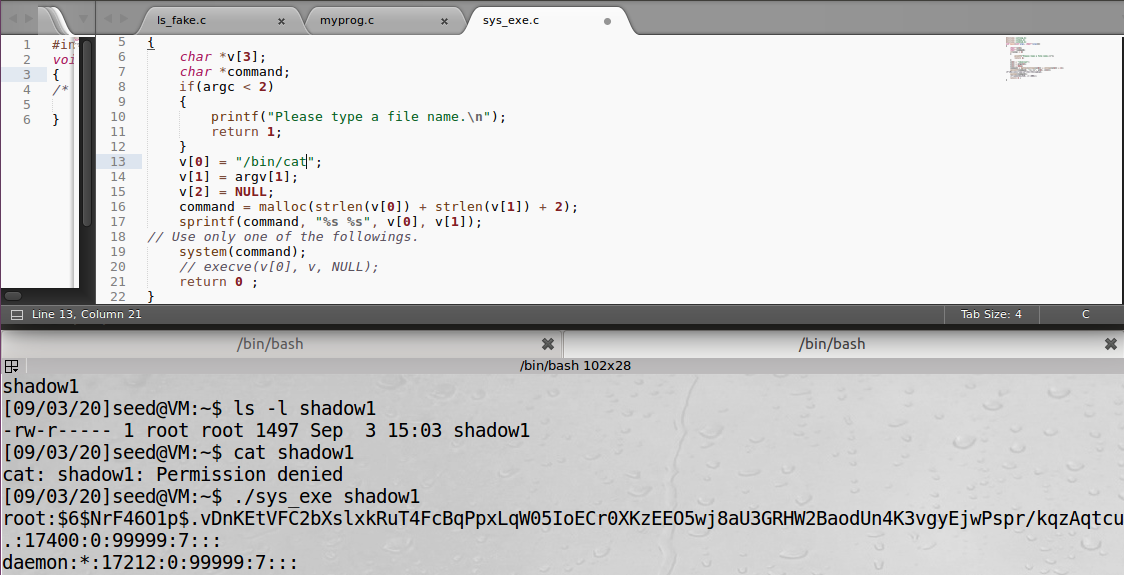




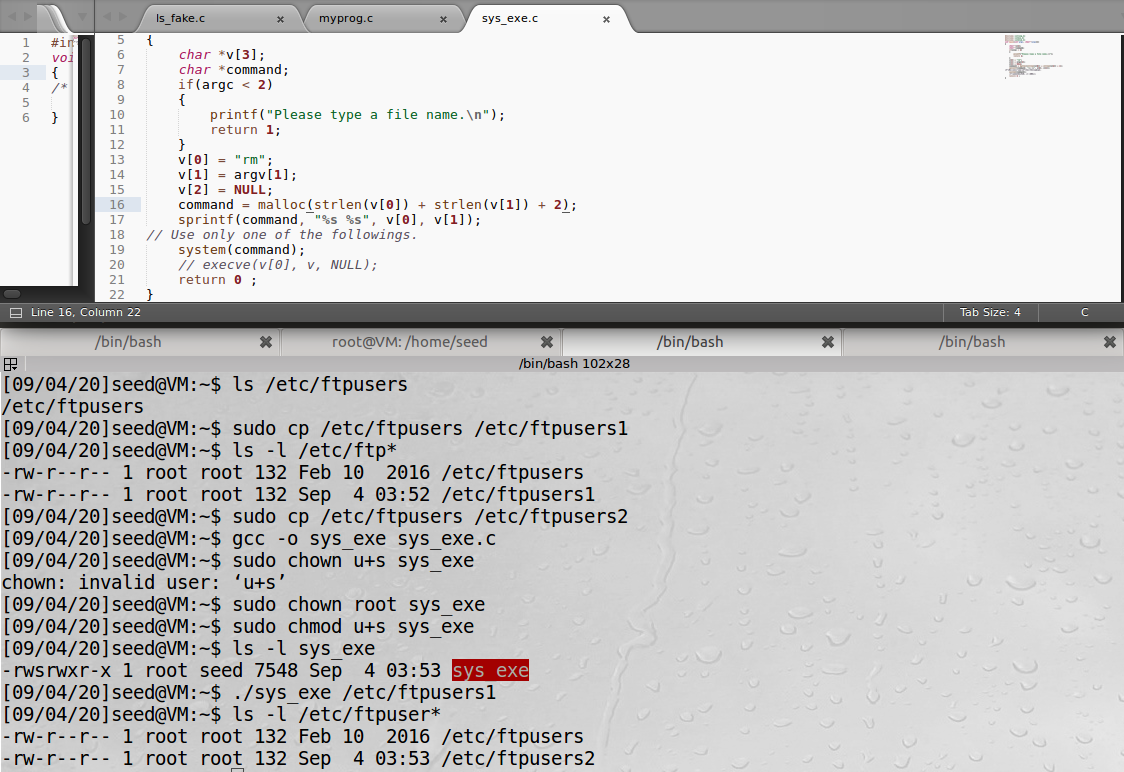
The child process didn’t inherit the LD \* environment variables when it’s a set-uid root program.

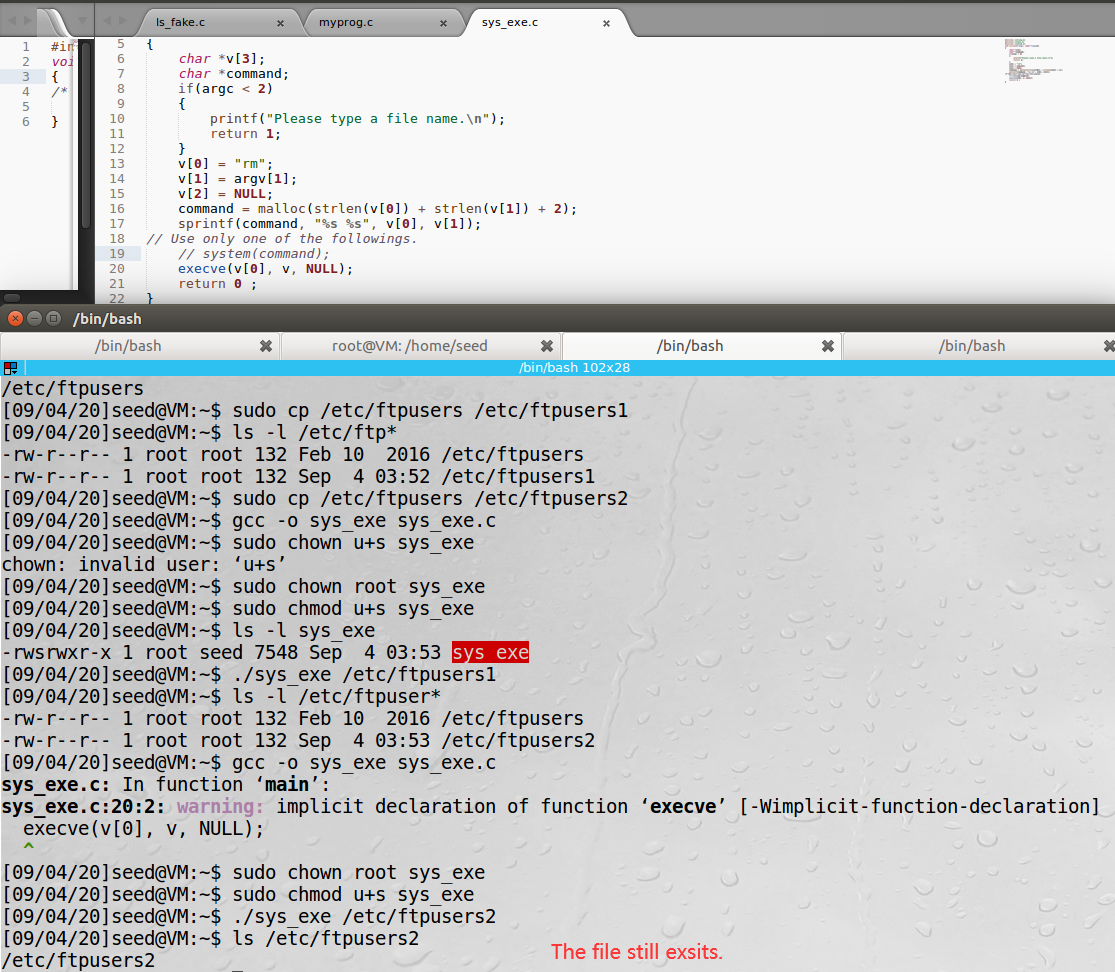
The first and third experiments with the MyProg program have seed user permissions, and the link library is also added in the LD\_PRELOAD environment variable of the SEED user.

## Task 8: Invoking External Programs Using system() versus execve()

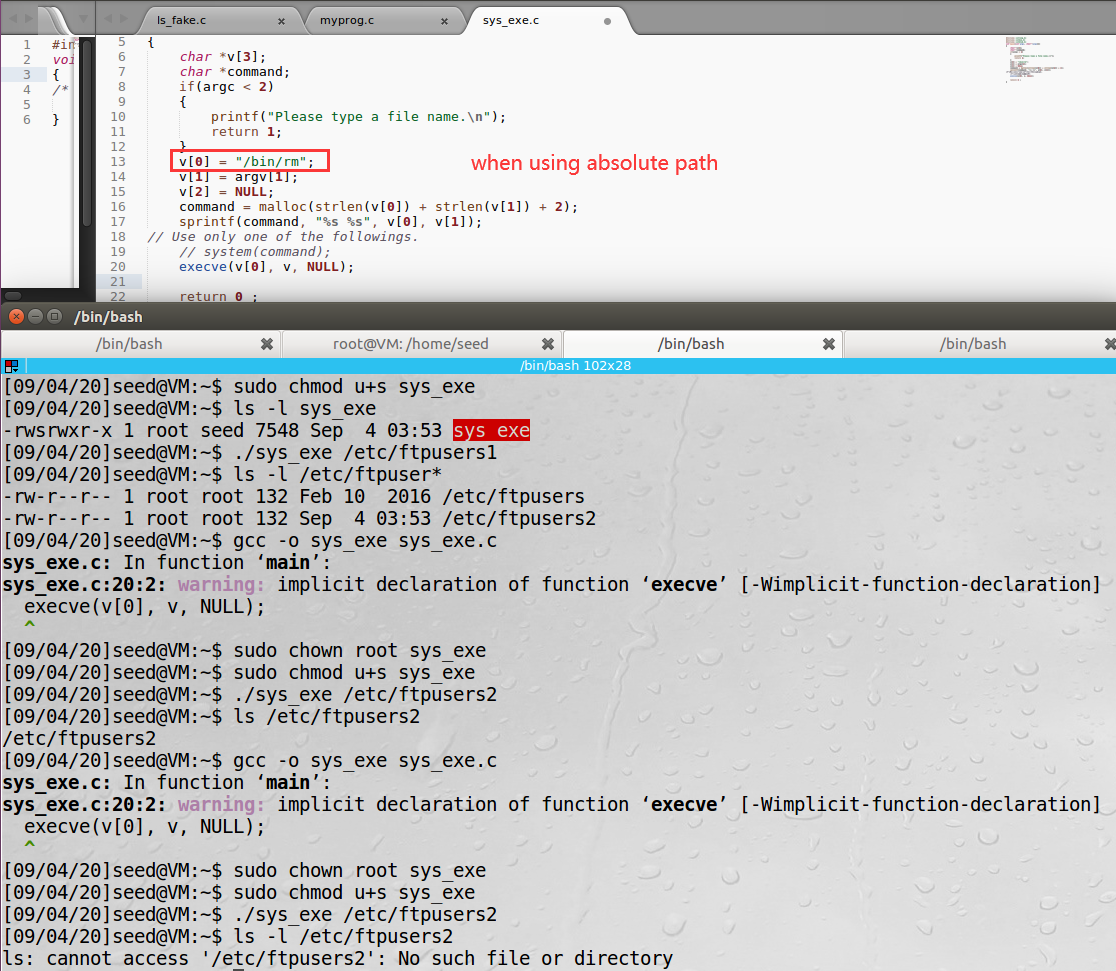


Change cat to rm.





The attack in step 1 didn’t work when using syscve.



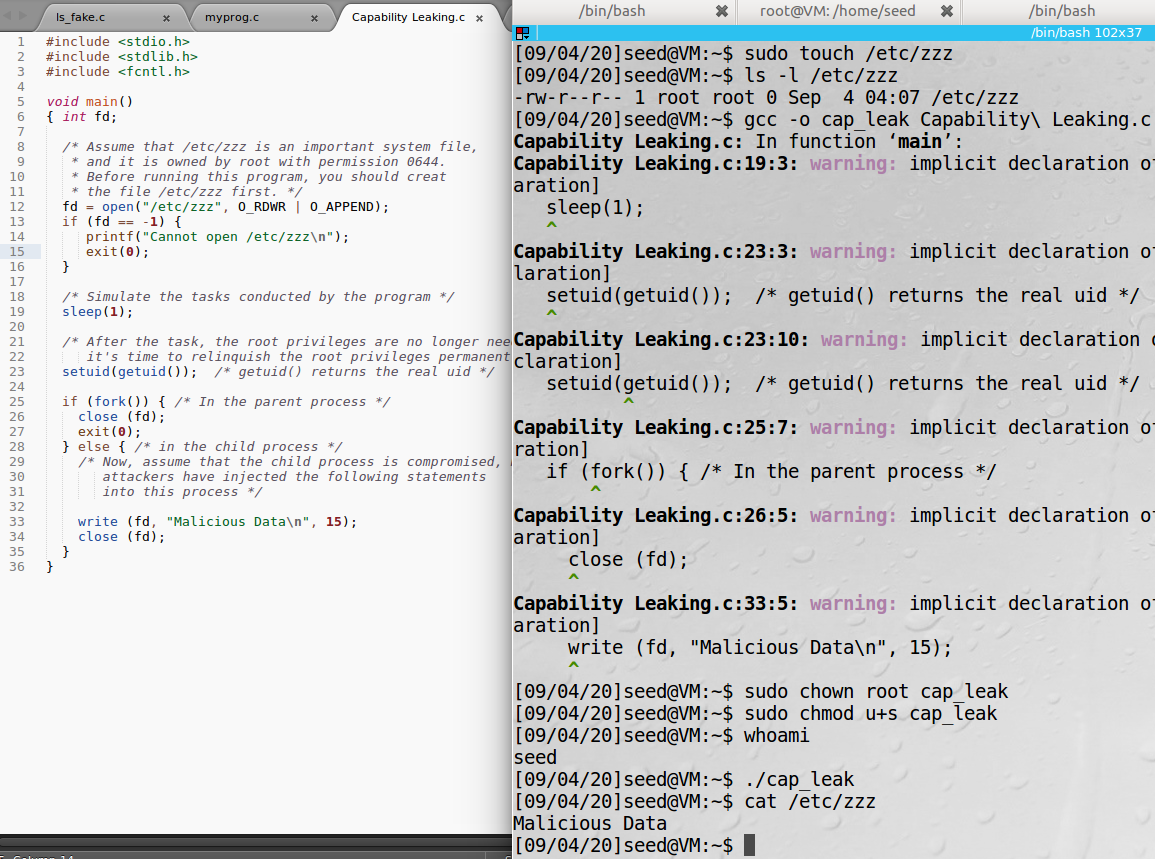
The attack in step 1 worked when using syscve and changing rm to /bin/rm.

system will call the shell (sh) to execute the command sent as an argument. The problem with system because the shell behavior depends on the user who run the command.

execve does not call a shell. It executes the program that passed to it as first argument. The program must be a binary executable or a script start with shebang line.

Of course both are dangerous depending on what is being executed when the process has root privileges. system(), though, brings some extra dangers due to the additional shell "layer" it uses that opens room security breaches as it invokes a root shell as in the case of your question

## Task 9: Capability Leaking



The file /etc/zzz was modified because the process is still privileged because it possesses privileged capabilities.