

Task 1: Manipulating Environment Variables

(1) 使用 printenv 命令输出环境变量输入命令 printenv，得到如下结果（截取部分）：

```

DG VTNR=7
RBIT SOCKETDIR=/tmp/orbit-seed
DG SESSION ID=c1
DG GREETER DATA DIR=/var/lib/lightdm-data/seed
BUS DISABLE SNOOPER=1
ERMINATOR UUID=urn:uuid:66c7962d-1afa-49a3-bb9c-c5e5c6027a8a
LUTTER IM MODULE=xim
SESSION=ubuntu
ITO LAUNCHED DESKTOP FILE PID=2628
NDROID HOME=/home/seed/android/android-sdk-linux
PG AGENT INFO=/home/seed/.gnupg/S.gpg-agent:0:1
ERM=xterm
DG MENU PREFIX=gnome-
HELL=/bin/bash
ERBY HOME=/usr/lib/jvm/java-8-oracle/db
IT LINUX ACCESSIBILITY ALWAYS ON=1
D PRELOAD=/home/seed/lib/boost/libboost_program_options.so.1.64.0:/home/seed/lib/boost/libboost_filesystem.s
i.1.64.0:/home/seed/lib/boost/libboost_system.so.1.64.0
IWINDOWID=25165828
IPSTART SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1449
NOME KEYRING CONTROL=
ITK MODULES=gail:atk-bridge:unity-gtk-module
ISER=seed
S 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:s
=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:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=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;3
*.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:
.jpg=01;35:*.jpeg=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;
5:*.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:*.

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
HOME=/home/seed
XDG SEAT=seat0
LANGUAGE=en_US
LIBGL_ALWAYS_SOFTWARE=1
GNOME DESKTOP SESSION_ID=this-is-deprecated
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-oBIkgWrrkN
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/ire
```

使用 printenv 命令查看 PATH 环境变量，得到如下结果：

```
[07/06/21] seed@VM:~$ printenv PWD
/home/seed
```

(2) 使用 export 和 unset 设置或删除环境变量

使用 export 设置环境变量，使用 echo 显示，\$ 符号实际作用是将变量转换成字符，方便输

```
[07/06/21] seed@VM:~$ export demo="Home/bin"
```

```
[07/06/21] seed@VM:~$ echo $demo
```

```
Home/bin
```

使用 unset 删除环境变量。

```
[07/06/21] seed@VM:~$ unset demo
```

```
[07/06/21] seed@VM:~$ echo $demo
```

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

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

(1) 编译 C 文件，将结果保存为 a.out 文件将代码保存为 demo.c 文件并放在桌面。进入桌面路径，编译 C 文件。

```
[07/06/21]seed@VM:~$ cd Desktop/  
[07/06/21]seed@VM:~/Desktop$ gcc demo.c
```

执行保存结果的 a.out 文件，查看代码的运行结果，发现为各个环境变量的值（截取部分）。

```
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  
HOME=/home/seed  
XDG_SEAT=seat0  
LANGUAGE=en_US  
LIBGL_ALWAYS_SOFTWARE=1  
GNOME_DESKTOP_SESSION_ID=this-is-deprecated  
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-oBIkgWrrkN  
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
```

(2) 按题意，将 child process 中 printenv()注释，将 process 中 parent printenv()取消注释，重新保存编译 C 文件。

执行保存结果的 b.out 文件，查看代码的运行结果，发现为各个环境变量的值（截取部分）。

```
COMPIZ_CONFIG_PROFILE=ubuntu-lowgfx  
IM_CONFIG_PHASE=1  
GDMSESSION=ubuntu  
SESSIONTYPE=gnome-session  
GTK2_MODULES=overlay-scrollbar  
SHLVL=1  
HOME=/home/seed  
XDG_SEAT=seat0  
LANGUAGE=en_US  
LIBGL_ALWAYS_SOFTWARE=1  
GNOME_DESKTOP_SESSION_ID=this-is-deprecated  
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-oBIkgWrrkN  
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  
=./b.out
```

(3) 比较两者结果

将 a.out、b.out 文件的结果分别保存为 child 和 parent 文件，再使用 diff 命令比较，发现两者除了文件名外完全相同。这说明子进程环境变量会继承父环境变量。进一步查阅资料了解到，子进程自父进程继承到进程的资格、环境、堆栈、内存等，但子进程所独有

的是不同的父进程号、自己的文件描述符和目录流的拷贝、在 tms 结构中的系统时间、不继承异步输入和输出等

```
[07/06/21]seed@VM:~/Desktop$ a.out > child
[07/06/21]seed@VM:~/Desktop$ b.out > parent
[07/06/21]seed@VM:~/Desktop$ diff child parent
49c49
< _=./a.out
---
> _=./b.out
```

Task 3: Environment variables and execve()

(1) 编译并运行以下程序。描述观察到的实验结果。该程序简单地调用了/usr/bin/env,该系统调用能够打印出当前进程的环境变量。

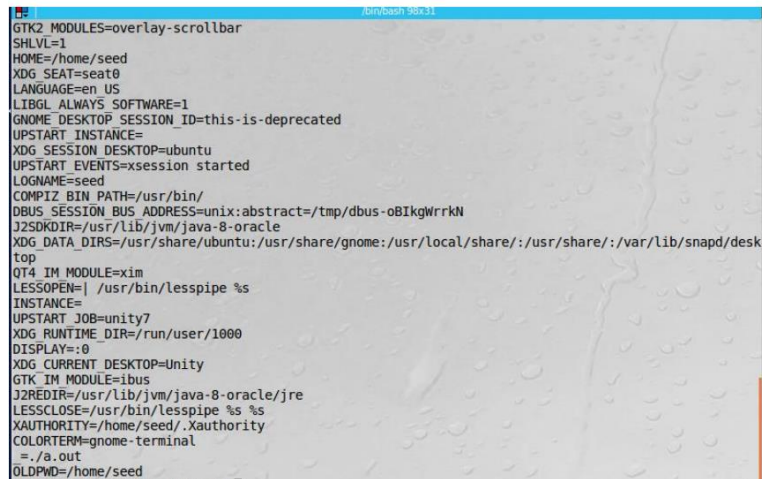
重新保存和编译文件，发现执行结果为空。

```
[07/06/21]seed@VM:~/Desktop$ gcc demo.c
[07/06/21]seed@VM:~/Desktop$ ./a.out
[07/06/21]seed@VM:~/Desktop$
```

查询函数 execve()的作用，其调用格式如下：int execve(const char * filename, char * const argv[], char * const envp[]) 第一个参数为一个可执行的有效路径名。第二个参数系利用数组指针来传递给执行文件，argv 是要调用的程序执行的参数序列，也就是我们要调用的程序需要传入的参数。envp 则为传递给执行文件的新环境变量数。所以在此处，我们赋予新进程的环境变量为空，自然印出环境变量结果为空。

(2) 把 execve () 的调用改为以下内容，观察结果

将原语句换为：execve("/usr/bin/env", argv, environ); 重新保存和编译文件，得到如下结果。



```
GTK2_MODULES=overlay-scrollbar
SHLVL=1
HOME=/home/seed
XDG_SEAT=seat0
LANGUAGE=en_US
LIBGL_ALWAYS_SOFTWARE=1
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
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-oBIkgWrrkN
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
_=./a.out
OLDPWD=/home/seed
```

(3) 描述实验结论

从以上实验可以看出，execve()产生的新进程的环境变量又调用时重新赋予，而 fork()则是直接继承父进程环境变量。

Task 4: Environment variables and system()

重新保存和编译文件


```
[07/06/21]seed@VM:~/Desktop$ gcc -o r4.out demo.c
[07/06/21]seed@VM:~/Desktop$ r4.out
```

得到如下结果（截取部分）：

```
;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=:1.93
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
GJS_DEBUG_OUTPUT=stderr
QT_IM_MODULE=ibus
PWD=/home/seed/Desktop
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:~/Desktop$ █
```

查阅资料得 system () 的调用格式如下：

int system (const char * string)

system()会调用 fork()产生子进程，由子进程来调用/bin/sh -c string 来执行参数 string 字符串所代表的命令，此命令执行完后随即返回原调用的进程。在调用 system()期间 SIGCHLD 信号会被暂时搁置，SIGINT 和 SIGQUIT 信号则会被忽略。

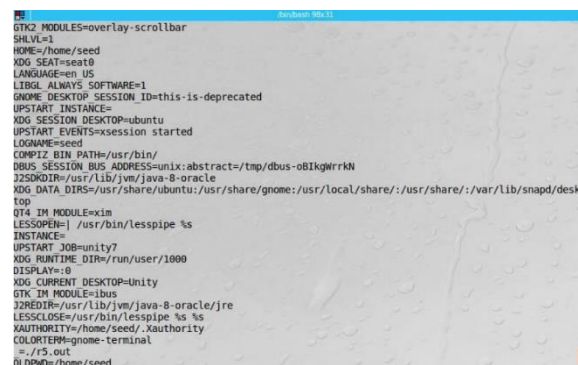
具体个描述为这样三个步骤：调用 fork () 函数新建一个子进程；在子进程中调用 exec 函数去执行 command；在父进程中调用 wait 去等待子进程结束。

返回值 =-1:出现错误 =0:调用成功但是没有出现子进程 >0:成功退出的子进程的 id 如果 system()在调用/bin/sh 时失败则返回 127，其他失败原因返回-1。若参数 string 为空指针 (NULL)，则返回非零值>。如果 system()调用成功则最后会返回执行 shell 命令后的返回值，但是此返回值也有可能为 system()调用/bin/sh 失败所返回的 127，因此最好能再检查 errno 来确认执行成功。

Task 5: Environment variable and Set-UID Programs

(1) 在当前进程中打印出所有的环境变量

重新保存、编译和执行给出的代码，得到如下结果（截取部分），此结果就是当前所有环境变量：



```
GTK2_MODULES=overlay-scrollbar
SHELL=1
HOME=/home/seed
XDG_SEAT=seat0
LANGUAGE=en_US
LIBGL_ALWAYS_SOFTWARE=1
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
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-oB1kgWrrkN
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
22REDIR=/usr/lib/jvm/java-8-oracle/jre
LESSCLOSE=/usr/bin/lesspipe %s %s
AUTHORITY=/home/seed/.xauthority
COLORTERM=gnome-terminal
= ./r5.out
OLDPWD=/home/seed
```

(2) 将上述程序的所有权改为 root，并使它成为一个 Set-UID 程序

先切换为 root 账户，使用 chown root:root demo.c 将此 c 文件权限改为 root 权限

```
[07/06/21]seed@VM:~/Desktop$ sudo chown root demo.c
[07/06/21]seed@VM:~/Desktop$ sudo chmod 4755 demo.c
```

(3) 使用一般用户登录终端，使用 export 命令设置如下环境变量：PATH 、

LD_LIBRARY_PATH、 ANY_NAME

export PATH="\$PATH:/usr/local/"

export LD_LIBRARY_PATH="\$LD_LIBRARY_PATH:/usr/local/"

export LXJ="/usr/local"

```
[07/06/21]seed@VM:~/Desktop$ export PATH="$PATH:/usr/local/"
[07/06/21]seed@VM:~/Desktop$ export LD_LIBRARY_PATH="$LD_LIBRARY_
PATH:/usr/local/"
[07/06/21]seed@VM:~/Desktop$ export LXJ="/usr/local"
[07/06/21]seed@VM:~/Desktop$ █
```

执行已经赋予 Set-UID 的 demo.c 程序，得到如下结果：

|LXJ=/usr/local

PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
:/usr/games:/usr/local/games:/snap/bin:./usr/local/

|LD_LIBRARY_PATH=:/usr/local/

可以看到，以上三个被定义的环境变量全部被包括在 shell 中。

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

保存代码为 demo6.c 文件，切换为 root 用户，将其编译为 demo6，并设置其所有者为 root， 赋予 SUID 特殊权限。

```
[07/06/21]seed@VM:~/Desktop$ sudo chown root:root demo6
[07/06/21]seed@VM:~/Desktop$ sudo chmod u+s demo6
```

使用 ls -l demo6 语句查看文件的权限，验证操作确实成功完成，符合题设条件

```
[07/06/21]seed@VM:~/Desktop$ sudo ls -l demo6
-rwsr-xr-x 1 root root 16696 Jul  6 05:08 demo6
```

将 bin/sh 复制到当前目录并命名为 ls，执行 demo6 就会获得 root 权限。详细分析，先看一下 PATH 环境变量，它的命令找寻顺序是先找寻当前目录，而当前目录我们自己编造了一个 ls，所以程序就会直接执行伪造的 ls。sh 原本的作用是创建一个新 shell，在执行此命令后我们会一直停留在子进程中，知道我们主动退出这个程序，我们才会回到原来的权限。

Task 8: Invoking external programs using system() versus execve()

(1) 编译上面的程序，把它的所有者变成它的所有者，并将它更改为一个 Set-UID 程序。该程序将使用 system () 来调用该命令。如果你是鲍勃，你能破坏系统的完整性吗？例

如，您能删除一个不能写入的文件吗？保存代码为 demo8.c 文件，切换为 root 用户，将其编译为 demo8，并设置其所有者为 root，赋予 SUID 特殊权限。

```
[07/06/21]seed@VM:~/Desktop$ sudo gcc -o demo8 demo.c
[07/06/21]seed@VM:~/Desktop$ sudo chown root:root demo8
[07/06/21]seed@VM:~/Desktop$ sudo chmod u+s demo8
```

新建一个名为 MY 的文件，并设置其权限为仅 root 用户可读、写、执行。

```
[07/06/21]seed@VM:~/Desktop$ touch MY
[07/06/21]seed@VM:~/Desktop$ sudo chmod u=rwx,g=---,o=--- MY
[07/06/21]seed@VM:~/Desktop$ ls -l MY
-rwx----- 1 seed seed 0 Jul  6 08:30 MY
```

执行 demo8，发现原本只有 root 用户才具有读、写、执行的 MY 文件，已经更名为 my

```
[07/06/21]seed@VM:~/Desktop$ demo8 "MY;mv MY my"
[07/06/21]seed@VM:~/Desktop$ ls
a.out  child  demo8  Labs_20.04  parent
b.out  demo6  demo.c  my          r4.out
```

(2) 注释掉 system(command)语句，并取消 execve () 语句;该程序将使用 execve () 来调用该命令。编译程序，并使之成为 Set-UID (由 root 拥有)。你在步骤 1 中的攻击仍然有效吗？请描述并解释你的观察。

重新编译 demo8 文件，并设置其所有者为 root，赋予 SUID 特殊权限。新建一个名为 MY 的文件，并设置其权限为仅 root 用户可读、写、执行。

```
[07/06/21]seed@VM:~/Desktop$ sudo gcc -o demo8 demo.c
[07/06/21]seed@VM:~/Desktop$ sudo chown root:root demo8
[07/06/21]seed@VM:~/Desktop$ sudo chmod u+s demo8
[07/06/21]seed@VM:~/Desktop$ touch MY
[07/06/21]seed@VM:~/Desktop$ sudo chmod u=rwx,g=---,o=--- MY
[07/06/21]seed@VM:~/Desktop$ ls -l MY
-rwx----- 1 seed seed 0 Jul  6 08:39 MY
```

执行 demo8，发现上一步的攻击方法已经失效

```
[07/06/21]seed@VM:~/Desktop$ ls
a.out  child  demo8  Labs_20.04  parent
b.out  demo6  demo.c  MY          r4.out
```