网安实验报告 1

57119118 尤何毅

Task 1

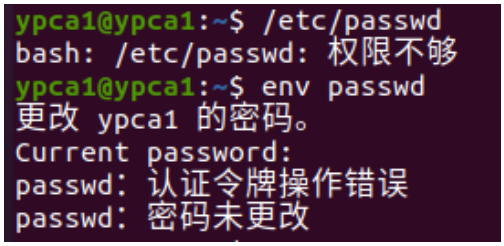
实验内容：

使用 printenv 或者 env 之类打印环境变量。
使用 export 和 unset 指令设置或者删除环境变量。

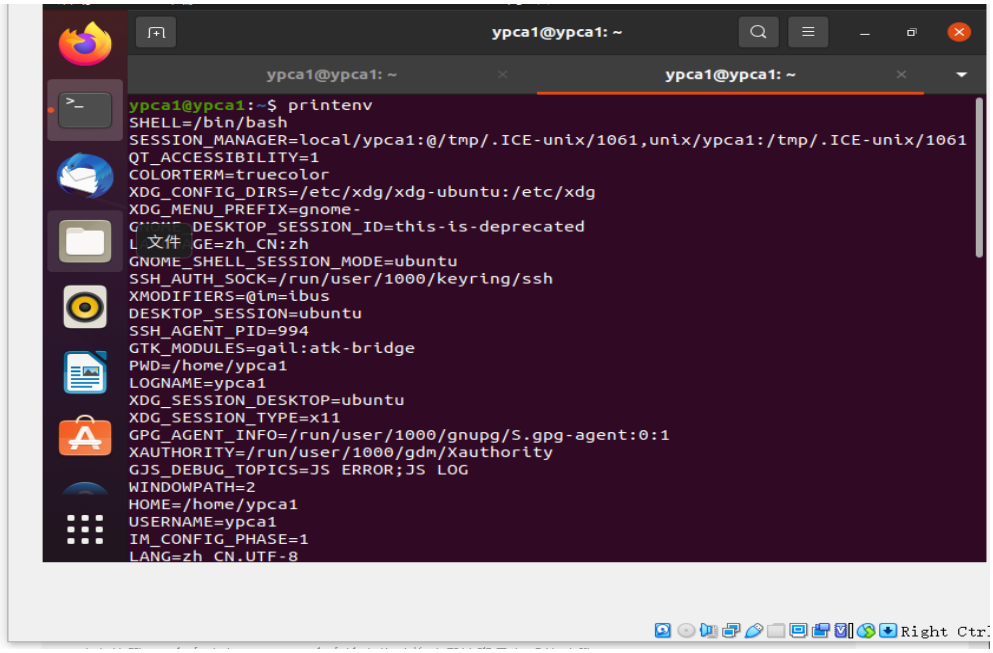
实验目的：

学习有关环境变量设置和删除的指令。

运行结果：



printenv 指令成功打印环境变量。



```
ypca1@ypca1:~$ TESTVAR="this is a test variable"
ypca1@ypca1:~$ echo $TESTVAR
this is a test variable
ypca1@ypca1:~$ printenv TESTVAR
this is a test variable
ypca1@ypca1:~$ export TESTVAR
ypca1@ypca1:~$ printenv TESTVAR
this is a test variable
ypca1@ypca1:~$ unset TESTVAR
ypca1@ypca1:~$ printenv TESTVAR
ypca1@ypca1:~$
```

使用 export 指令成功设置了一个环境变量 TESTVAR，并用 printenv 访问；之后用 unset 将其删除，再访问则打印不出值了。

实验体会：

使用 export 和 unset 可以直接对环境变量进行操作，十分方便。

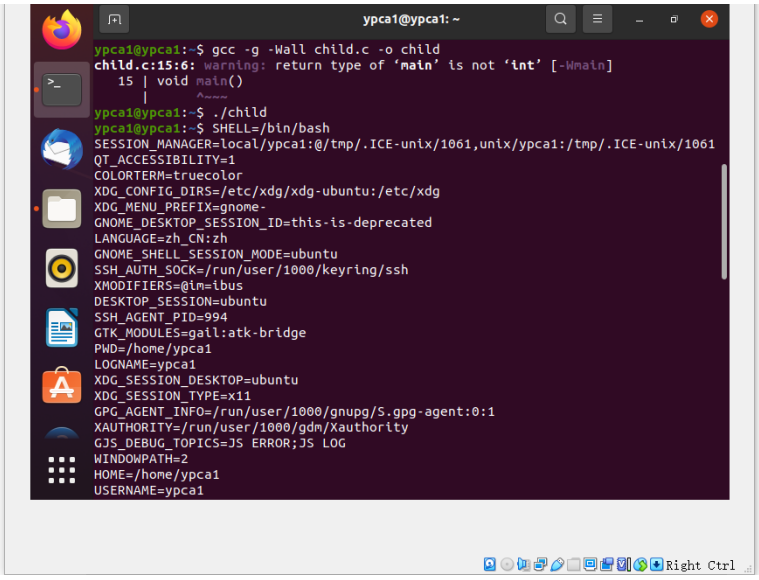
Task 2

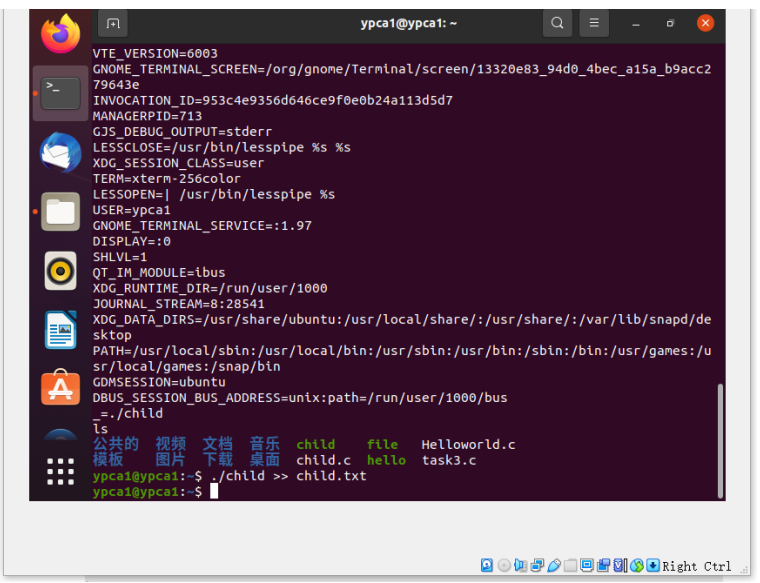
实验内容：编译已有程序并运行，再把输出结果存在一个文件中

实验目的：研究子进程如何从其父进程获取其环境变量

源程序：

步骤 1





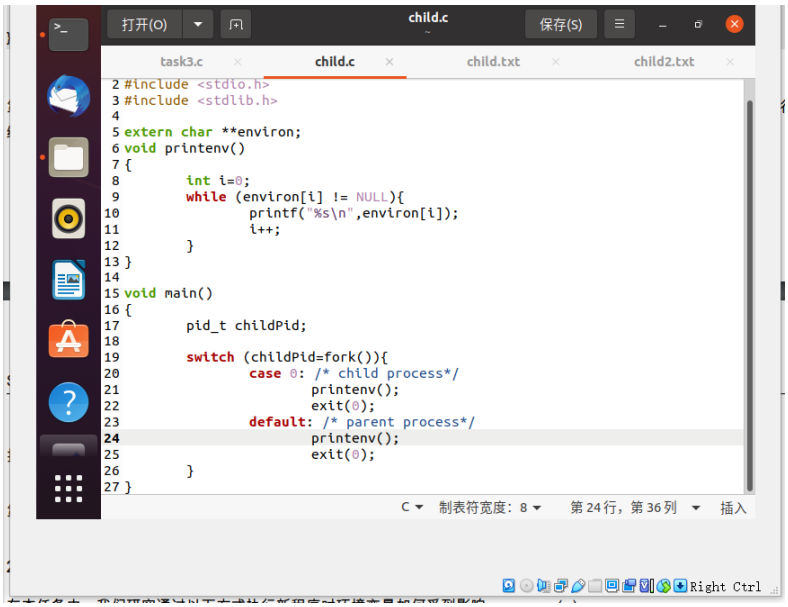
task3.c | child.c | child.txt

```c
#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);
        }
```

步骤 2

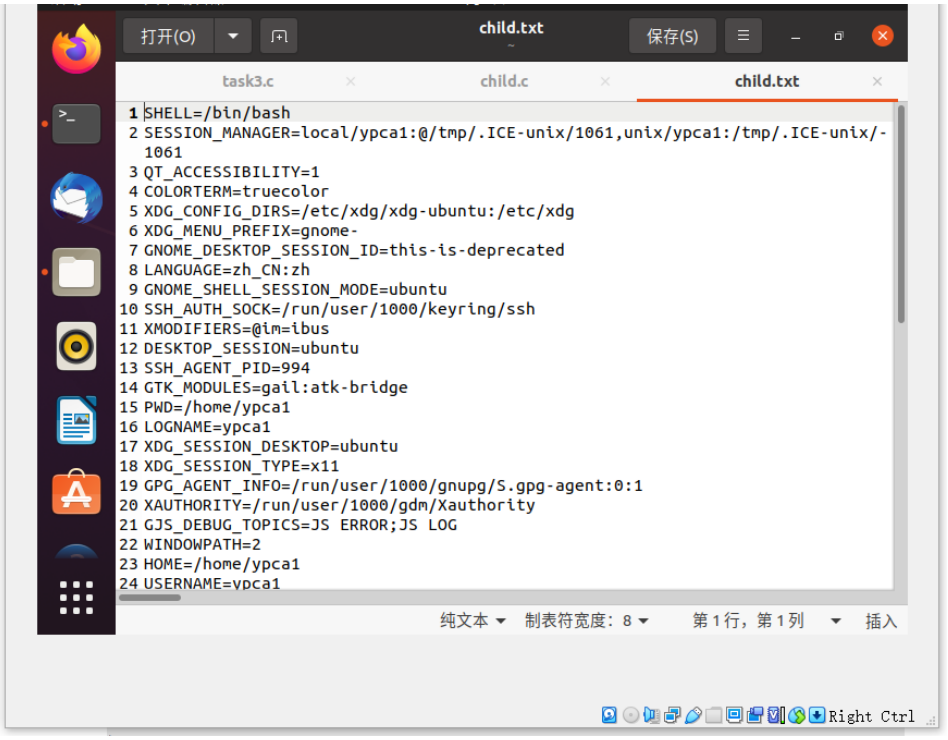
```
ypca1@ypca1:~$ gcc -g -Wall child.c -o child2
child.c:15:6: warning: return type of ‘main’ is not ‘int’ [-Wmain]
   15 | void main()
      |      ^~~~
ypca1@ypca1:~$ ./child2 >>child2.txt
ypca1@ypca1:~$
```

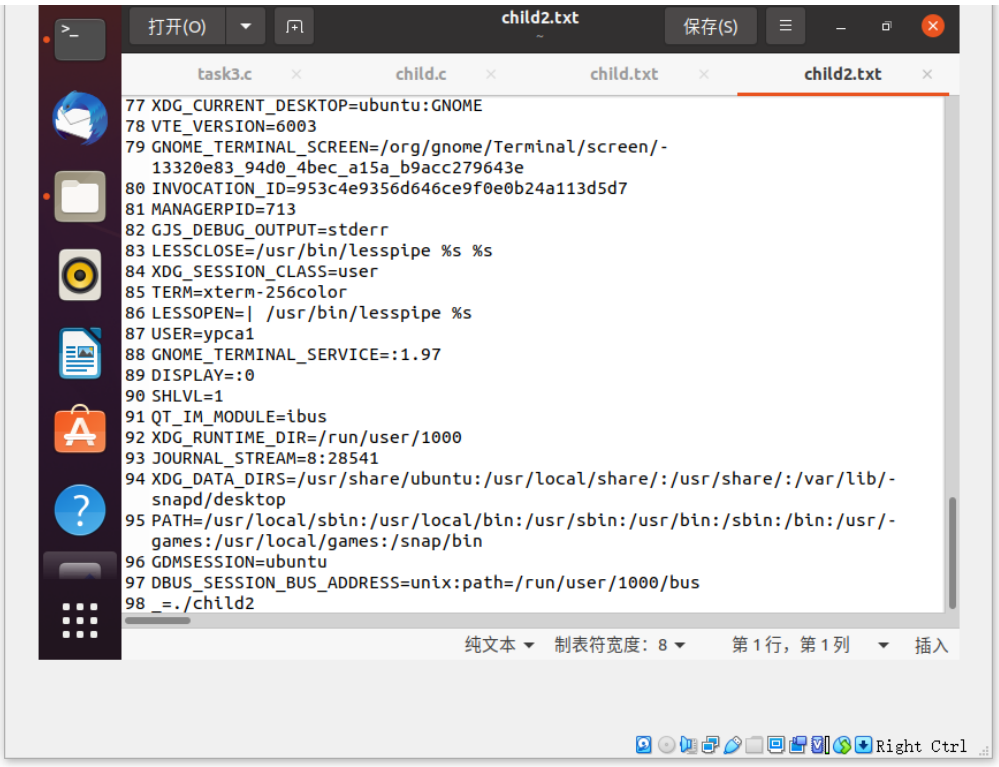
修改后的 child.c



运行结果：如图，成功存进了 child.txt 文件和 child2.txt 文件中

结果显示，新进程成功读取了环境变量，并且存储在一个 txt 文件中可以查看。最终结果显示，区别在于取消注释后输出了 parent process 中环境变量的值，而第一次输出到 child.txt 只有 child process 的环境变量。





实验体会：

编译初因为不清楚如何将编译的结果直接显示出来使用了一些时间进行网上搜索，最终得到解决。

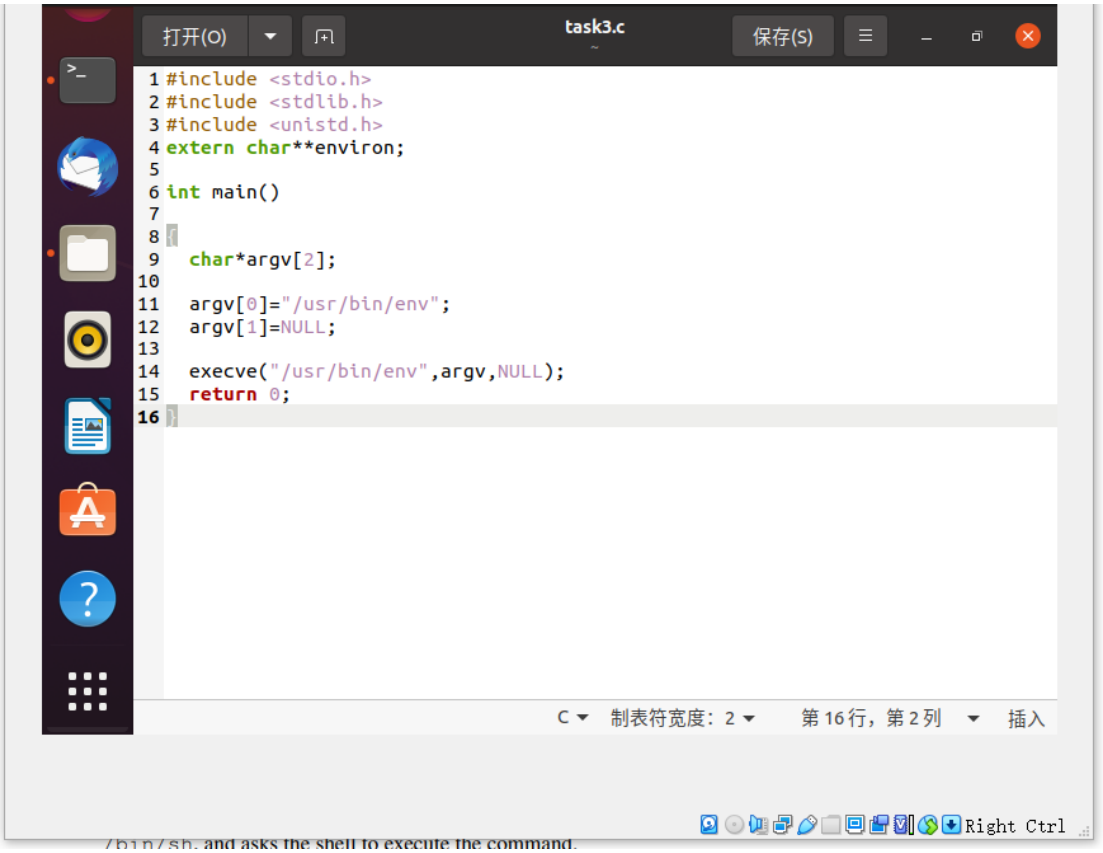
Task 3

实验内容：编译并运行程序，描述观察到的结果。

实验目的：了解环境变量和 execve()函数

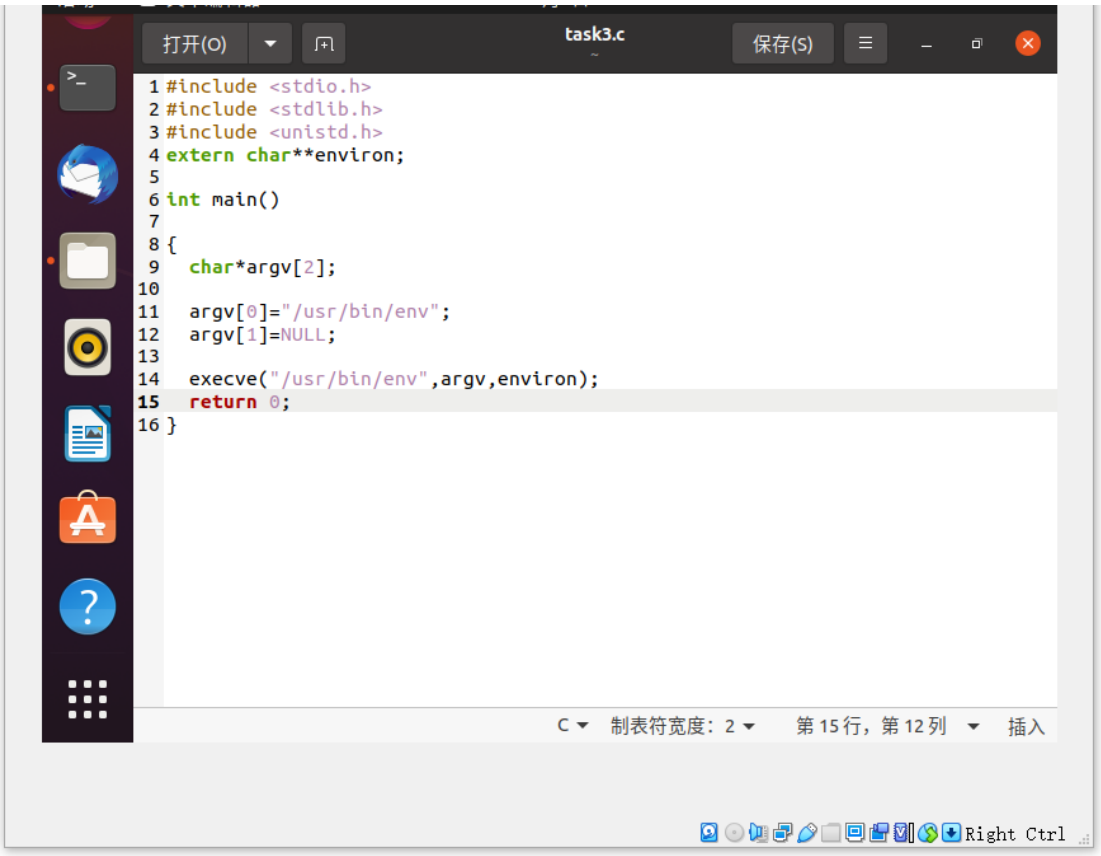
源程序：

更改前：

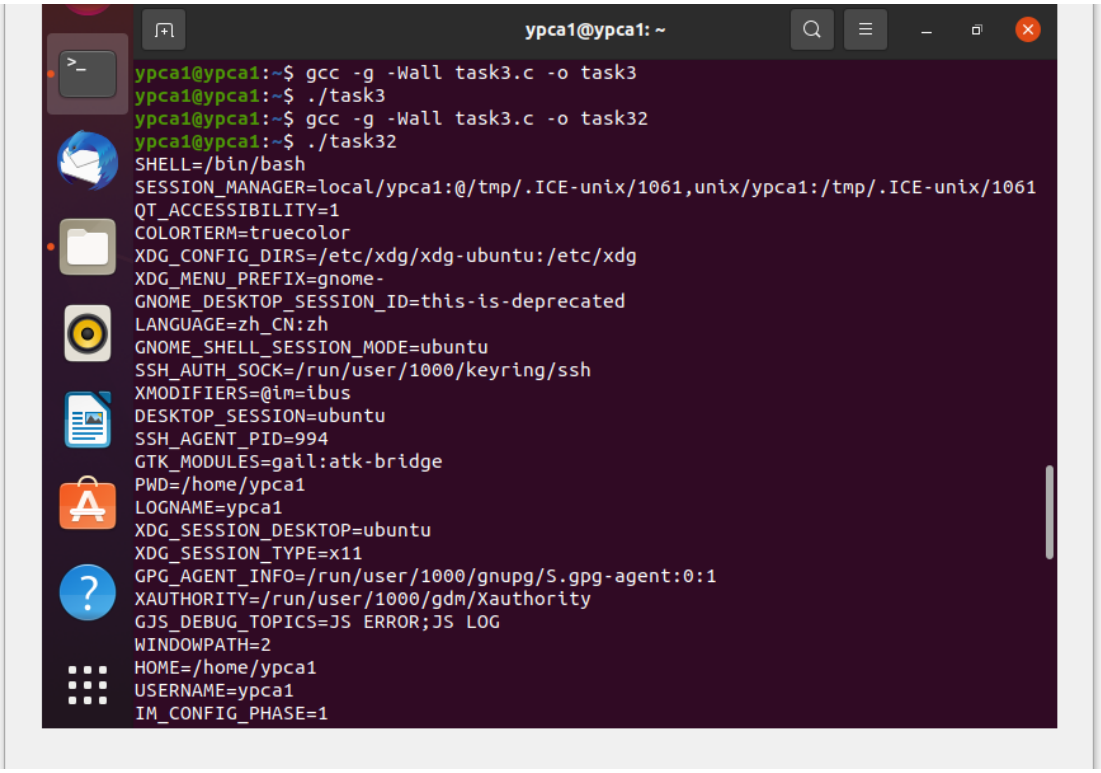


/bin/sh, and asks the shell to execute the command.

更改后：



运行结果：



结果表明，前一次未输出环境变量而后一次可以成功输出环境变量。

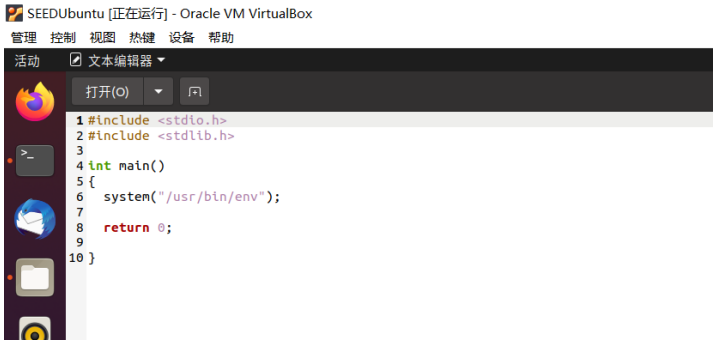
实验体会：刚开始没发现少了一个头文件，运行不成功，试了好几次才发现。

Task 4

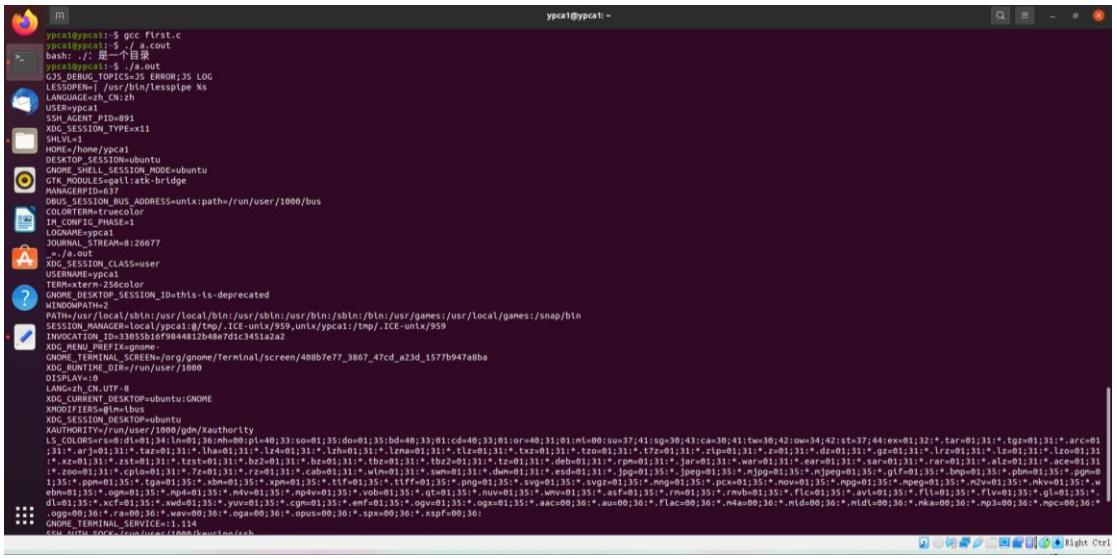
实验内容：编译并运行程序，描述观察到的结果。

实验目的：了解环境变量和 execve()函数

源程序：



运行结果：

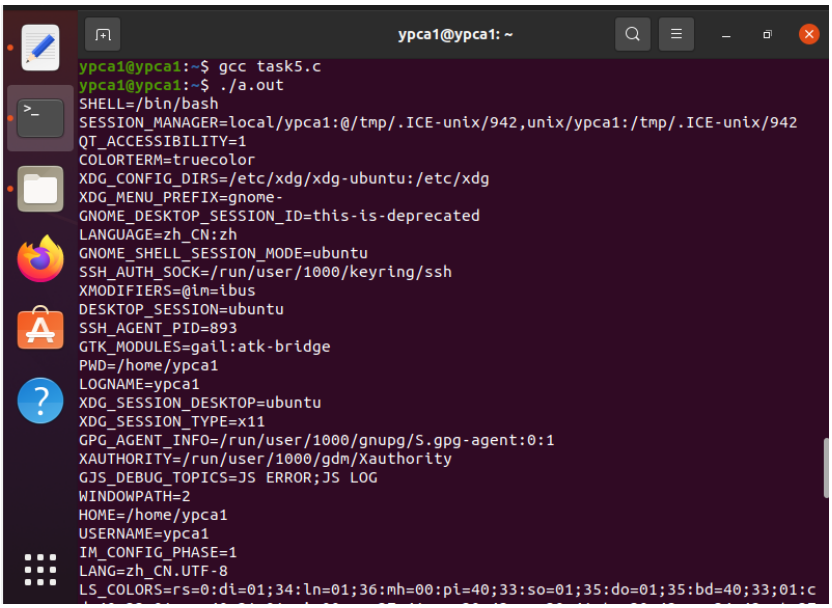


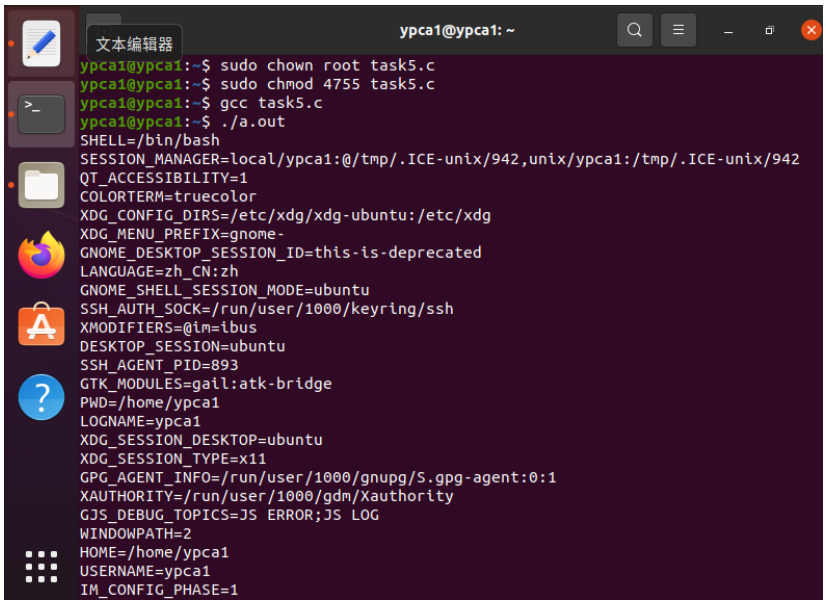
Task 5

实验内容： 了解 set-uid 程序

源程序：

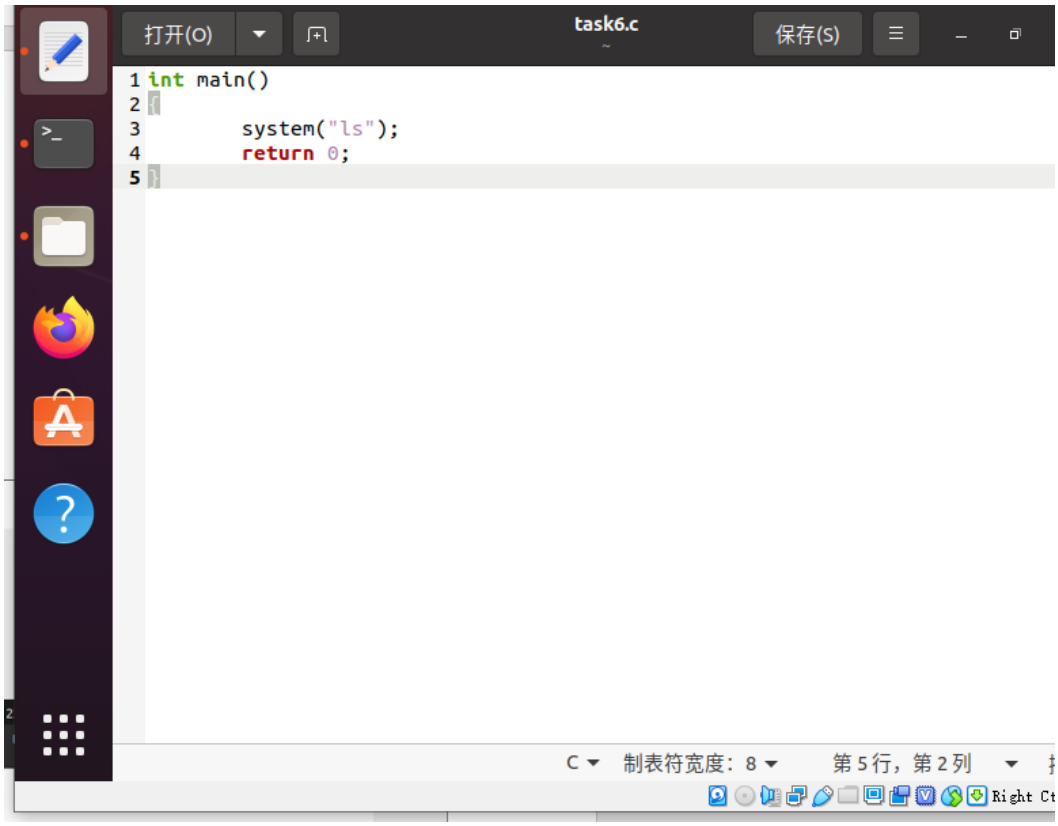
运行结果：



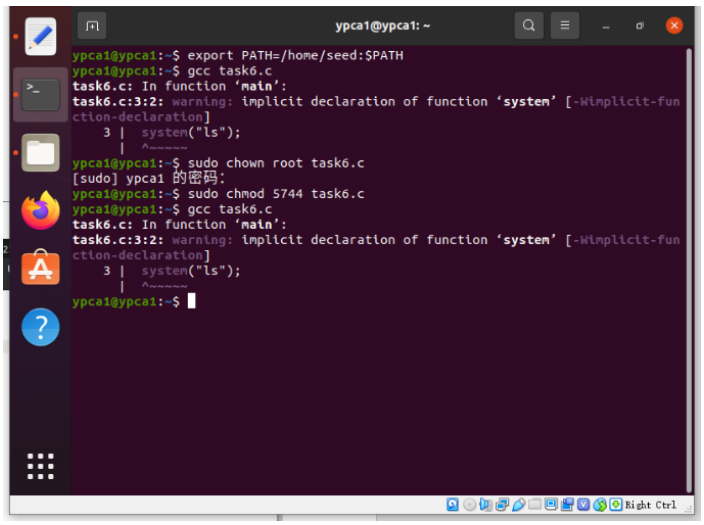

文本编辑器
ypca1@ypca1: ~
ypca1@ypca1:~$ sudo chown root task5.c
ypca1@ypca1:~$ sudo chmod 4755 task5.c
ypca1@ypca1:~$ gcc task5.c
ypca1@ypca1:~$ ./a.out
SHELL=/bin/bash
SESSION_MANAGER=local/ypca1:@/tmp/.ICE-unix/942,unix/ypca1:/tmp/.ICE-unix/942
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
LANGUAGE=zh_CN:zh
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=893
GTK_MODULES=gail:atk-bridge
PWD=/home/ypca1
LOGNAME=ypca1
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
XAUTHORITY=/run/user/1000/gdm/Xauthority
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
WINDOWPATH=2
HOME=/home/ypca1
USERNAME=ypca1
IM_CONFIG_PHASE=1

Task 6

源程序：


打开(O)
task6.c
保存(S)
1 int main()
2 {
3 system("ls");
4 return 0;
5 }
C
制表符宽度：8
第 5 行，第 2 列

运行结果：



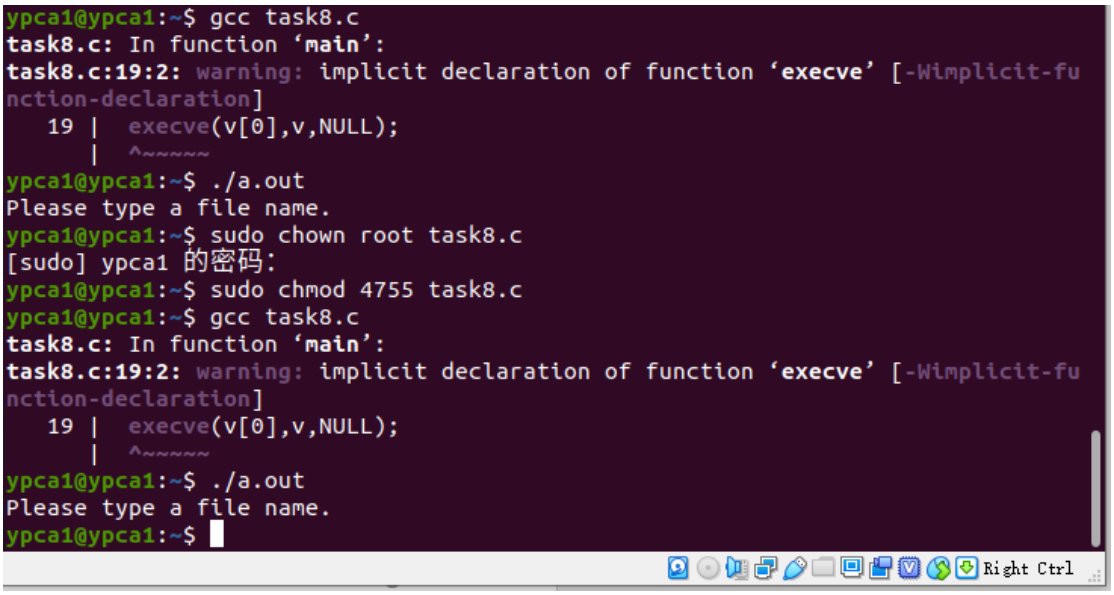
Task 8

源程序：

更改前

更改后

运行结果：


ypca1@ypca1:~$ gcc task8.c
task8.c: In function ‘main’:
task8.c:19:2: warning: implicit declaration of function ‘execve’ [-Wimplicit-function-declaration]
19 | execve(v[0],v,NULL);
| ^~~~~~
ypca1@ypca1:~$ ./a.out
Please type a file name.
ypca1@ypca1:~$ sudo chown root task8.c
[sudo] ypca1 的密码：
ypca1@ypca1:~$ sudo chmod 4755 task8.c
ypca1@ypca1:~$ gcc task8.c
task8.c: In function ‘main’:
task8.c:19:2: warning: implicit declaration of function ‘execve’ [-Wimplicit-function-declaration]
19 | execve(v[0],v,NULL);
| ^~~~~~
ypca1@ypca1:~$ ./a.out
Please type a file name.
ypca1@ypca1:~$
Right Ctrl