

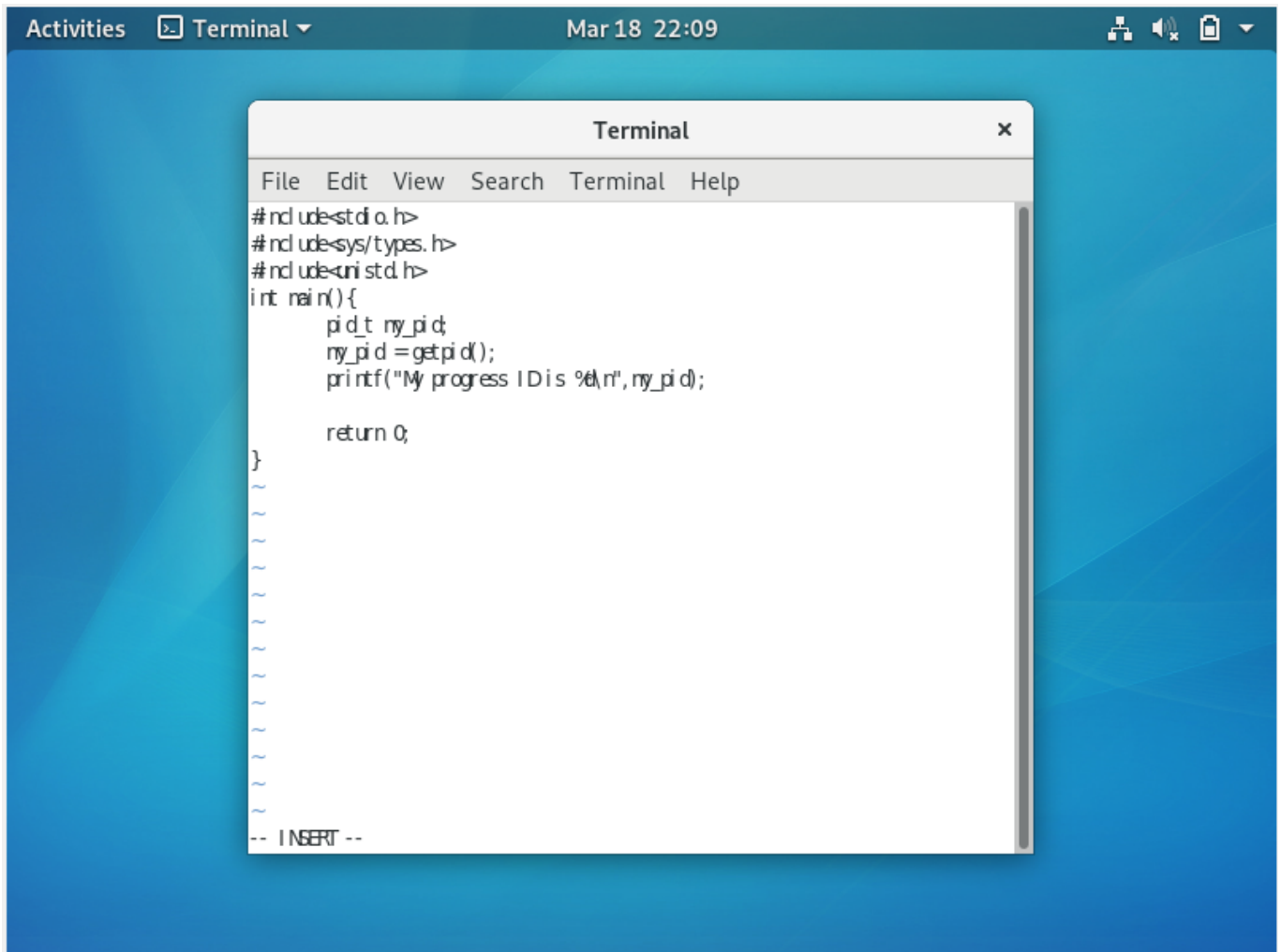
Linux环境下C程序编译

创建cpp程序并运行的的方法

1.打开虚拟机命令行 2.创建cpp文件: vi name.cpp 3.输入代码 4.保存并退出: 按ESC 输入: w 再输入: wq 5.编译代码: g++ name.cpp -o name 6.运行程序: ./name.cpp

获取进程的PID

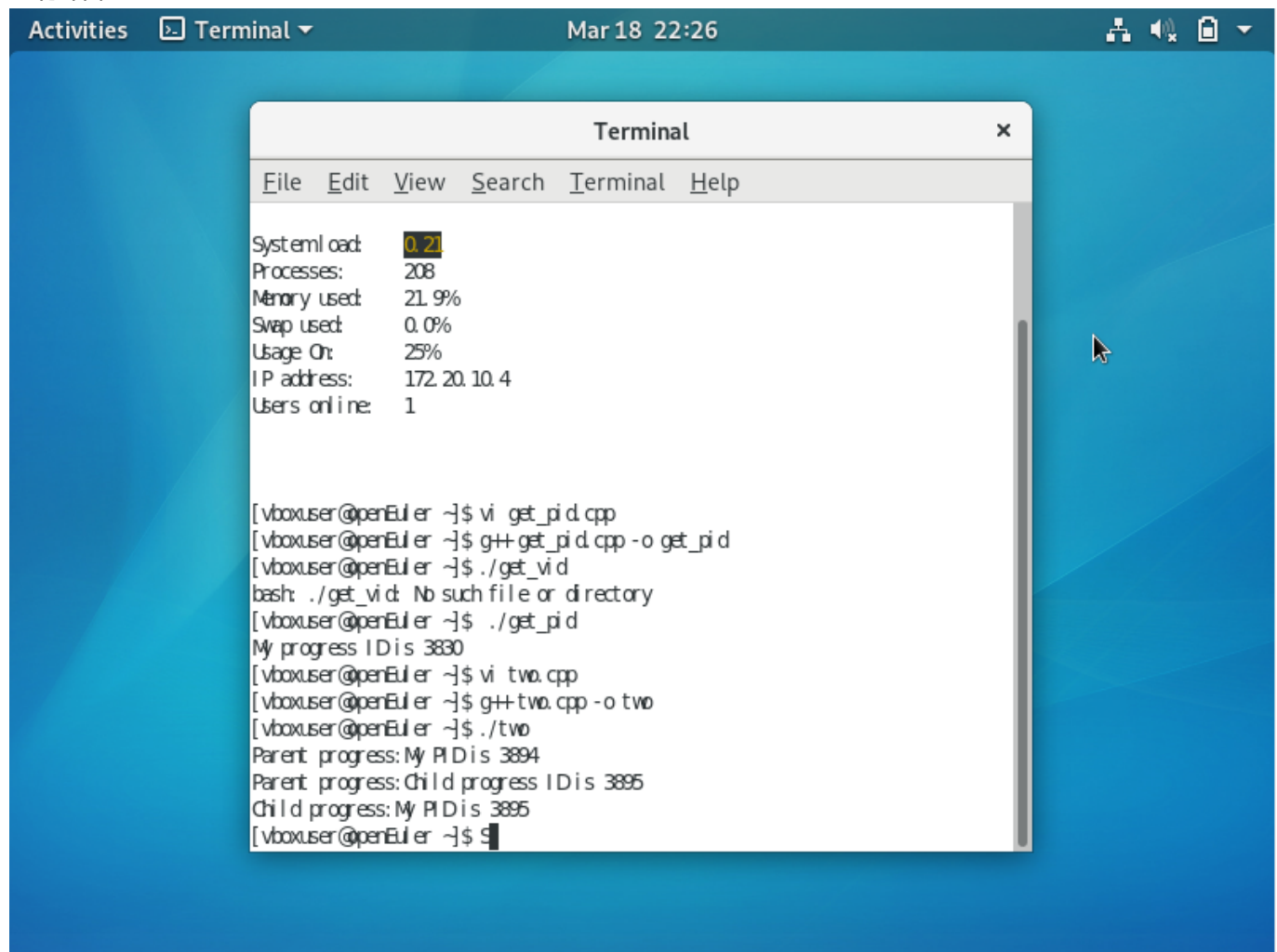
代码:

A screenshot of a Linux desktop environment with a blue background. In the foreground, a terminal window titled "Terminal" is open. The terminal has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The code displayed in the terminal is a C program to get the process ID (PID). The code includes headers for stdio.h, sys/types.h, and unistd.h. The main function declares a pid_t variable named my_pid, assigns it the value of getpid(), prints it with a format string, and then returns 0. The terminal shows the code being typed, with some lines starting with tilde (~) as placeholders. The status bar at the bottom of the terminal window shows "-- INSERT --".

```
File Edit View Search Terminal Help
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
int main(){
    pid_t my_pid;
    my_pid = getpid();
    printf("My process ID is %d\n", my_pid);

    return 0;
}
~
~
~
~
~
~
~
~
~
~
-- INSERT --
```

运行结果:



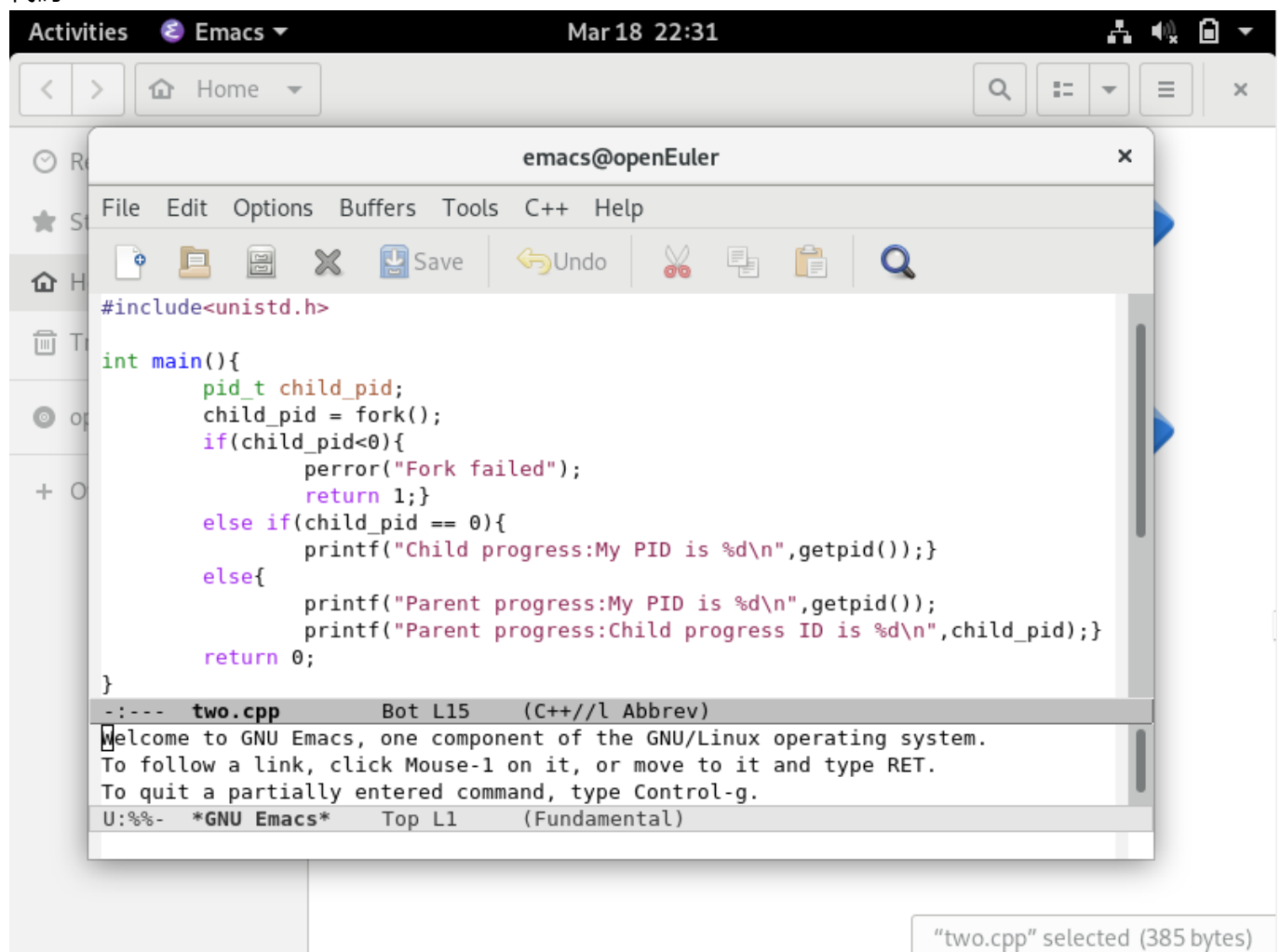
The screenshot shows a terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal output displays system statistics and the execution of several C++ programs. The system statistics include: System load: 0.21, Processes: 208, Memory used: 21.9%, Swap used: 0.0%, Usage On: 25%, IP address: 172.20.10.4, and Users online: 1. The command sequence shows the user editing and compiling 'get_pid.cpp' to 'get_pid', running it to get their own PID (3830), then editing and compiling 'two.cpp' to 'two', and running it to show parent and child PIDs (3894 and 3895).

```
System load: 0.21
Processes: 208
Memory used: 21.9%
Swap used: 0.0%
Usage On: 25%
IP address: 172.20.10.4
Users online: 1

[vboxuser@openEuler ~]$ vi get_pid.cpp
[vboxuser@openEuler ~]$ g++ get_pid.cpp -o get_pid
[vboxuser@openEuler ~]$ ./get_pid
bash: ./get_pid: No such file or directory
[vboxuser@openEuler ~]$ ./get_pid
My progress ID is 3830
[vboxuser@openEuler ~]$ vi two.cpp
[vboxuser@openEuler ~]$ g++ two.cpp -o two
[vboxuser@openEuler ~]$ ./two
Parent progress: My PID is 3894
Parent progress: Child progress ID is 3895
Child progress: My PID is 3895
[vboxuser@openEuler ~]$
```

获进程创建与父子进程关系实验

代码:



```
#include<unistd.h>

int main(){
    pid_t child_pid;
    child_pid = fork();
    if(child_pid<0){
        perror("Fork failed");
        return 1;}
    else if(child_pid == 0){
        printf("Child progress:My PID is %d\n",getpid());}
    else{
        printf("Parent progress:My PID is %d\n",getpid());
        printf("Parent progress:Child progress ID is %d\n",child_pid);}
    return 0;
}
```

two.cpp Bot L15 (C++//l Abbrev)

Welcome to GNU Emacs, one component of the GNU/Linux operating system.
To follow a link, click Mouse-1 on it, or move to it and type RET.
To quit a partially entered command, type Control-g.

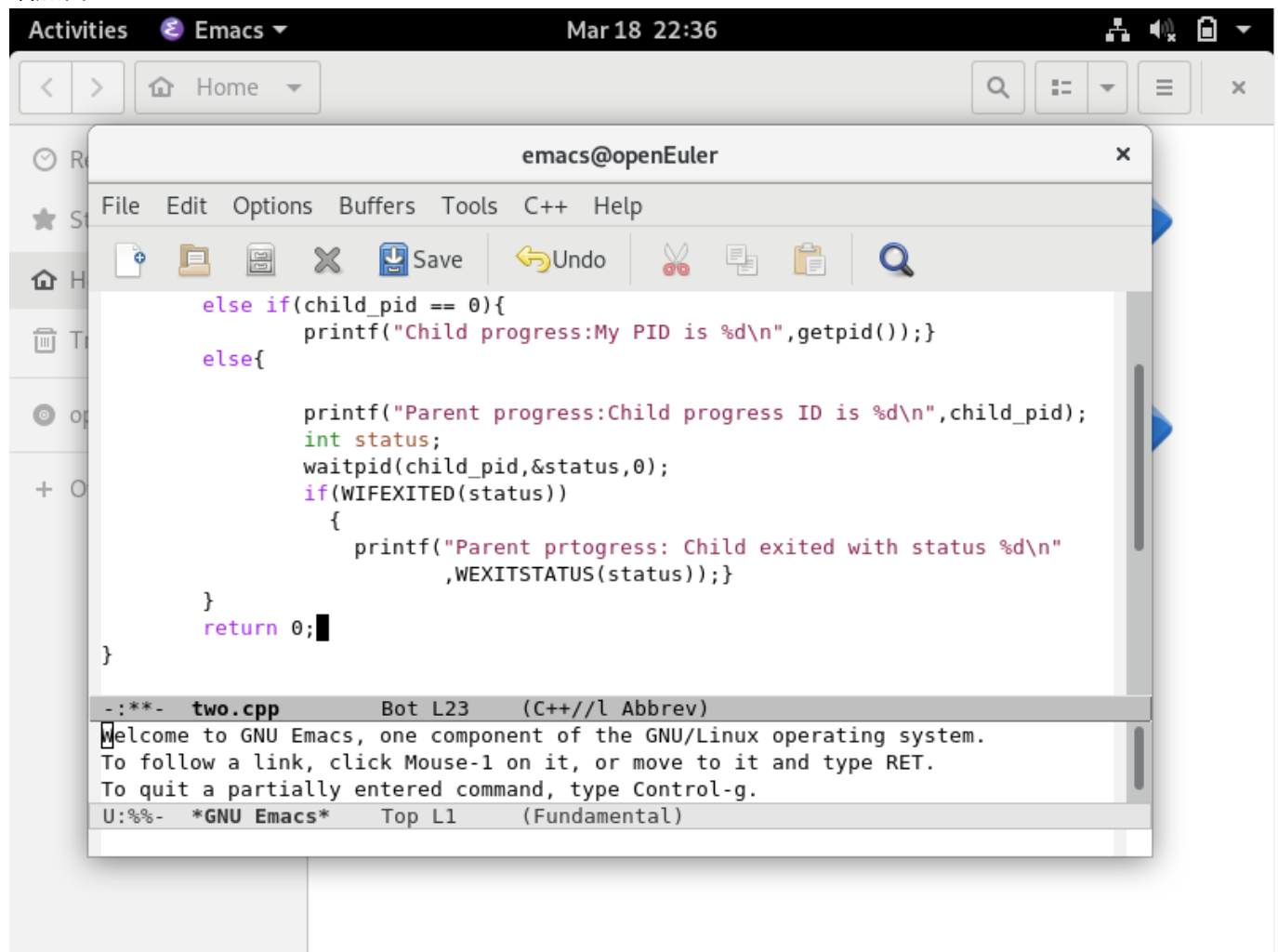
U:%%- *GNU Emacs* Top L1 (Fundamental)

"two.cpp" selected (385 bytes)

运行结果: Parent process: My PID is 70845 Parent process: Child process ID is 70846 Child process:My PID is 70846

父进程等待子进程退出测试

增加代码:



```
File Edit Options Buffers Tools C++ Help

else if(child_pid == 0){
    printf("Child progress:My PID is %d\n",getpid());}
else{

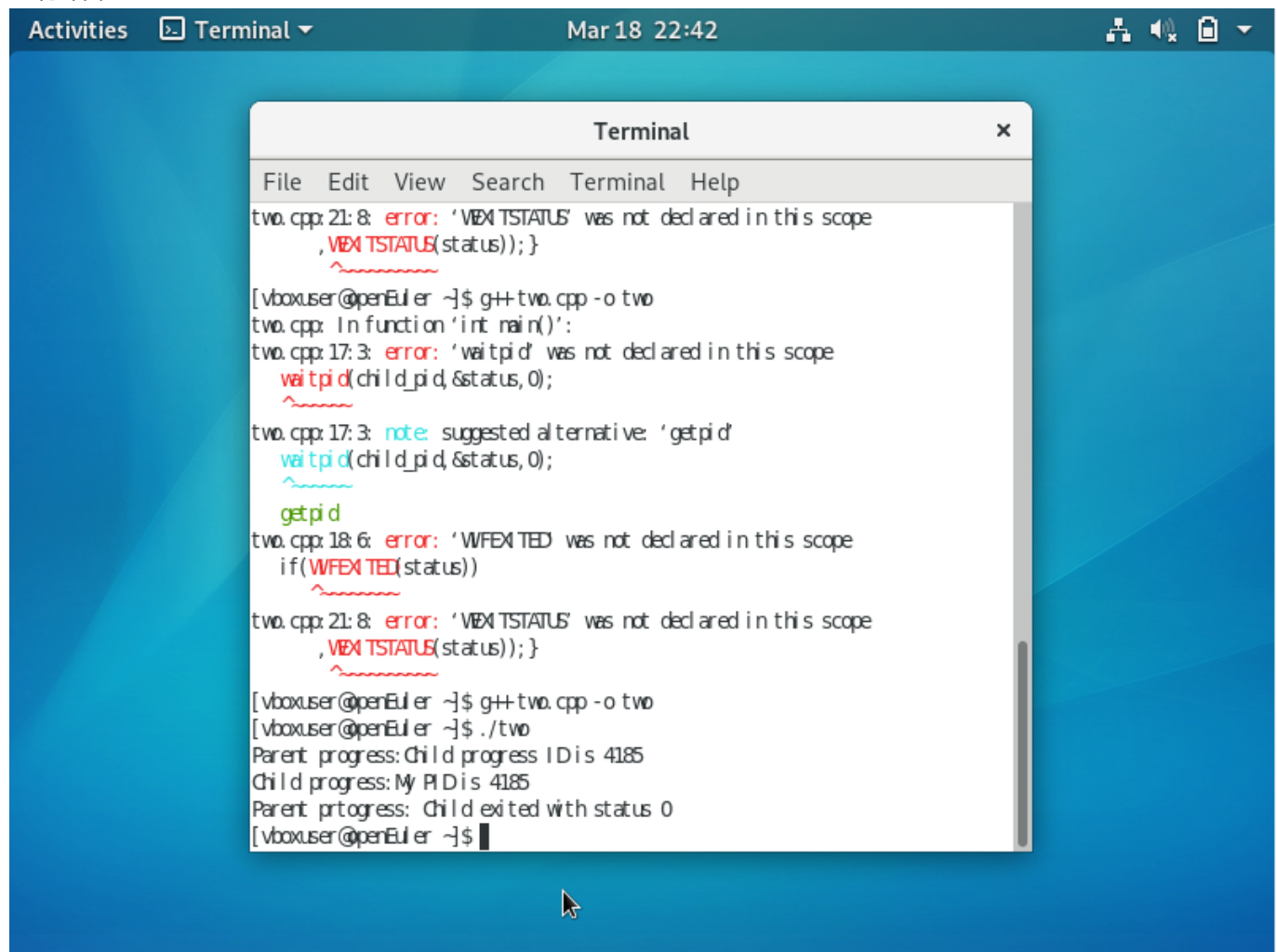
    printf("Parent progress:Child progress ID is %d\n",child_pid);
    int status;
    waitpid(child_pid,&status,0);
    if(WIFEXITED(status))
    {
        printf("Parent prtogress: Child exited with status %d\n"
            ,WEXITSTATUS(status));}
    }
    return 0;
}
```

two.cpp Bot L23 (C++//l Abbrev)

Welcome to GNU Emacs, one component of the GNU/Linux operating system.
To follow a link, click Mouse-1 on it, or move to it and type RET.
To quit a partially entered command, type Control-g.

U:%%- *GNU Emacs* Top L1 (Fundamental)

运行结果:



```
Activities Terminal Mar 18 22:42

Terminal
File Edit View Search Terminal Help

two.cpp: 21: 8: error: 'WEXITSTATUS' was not declared in this scope
      , WEXITSTATUS(status)); }
      ^~~~~~

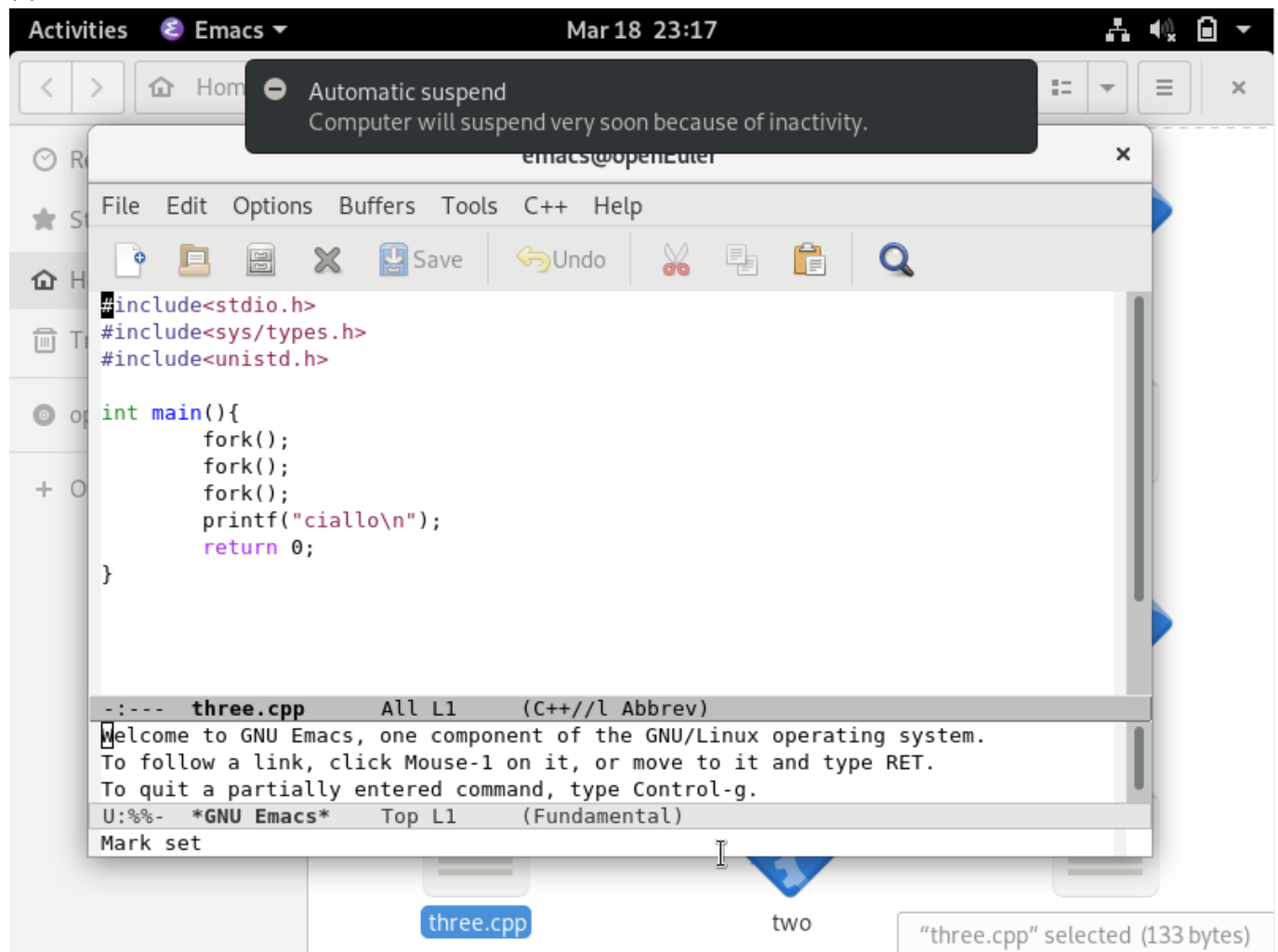
[vboxuser@openEuler ~]$ g++ two.cpp -o two
two.cpp: In function 'int main()':
two.cpp: 17: 3: error: 'waitpid' was not declared in this scope
      waitpid(child_pid, &status, 0);
      ^~~~~~
two.cpp: 17: 3: note: suggested alternative: 'getpid'
      waitpid(child_pid, &status, 0);
      ^~~~~~
      getpid
two.cpp: 18: 6: error: 'WIFEXITED' was not declared in this scope
      if(WIFEXITED(status))
      ^~~~~~
two.cpp: 21: 8: error: 'WEXITSTATUS' was not declared in this scope
      , WEXITSTATUS(status)); }
      ^~~~~~

[vboxuser@openEuler ~]$ g++ two.cpp -o two
[vboxuser@openEuler ~]$ ./two
Parent progress: Child progress ID is 4185
Child progress: My PID is 4185
Parent progress: Child exited with status 0
[vboxuser@openEuler ~]$
```

父进程在调用 `waitpid()` 后进入等待状态，直至子进程退出后才继续执行后续代码。子进程正常退出（退出状态为 0），并且父进程通过 `WIFEXITED` 与 `WEXITSTATUS` 检查子进程的退出状态

多次 `fork()` 进程创建实验

代码:

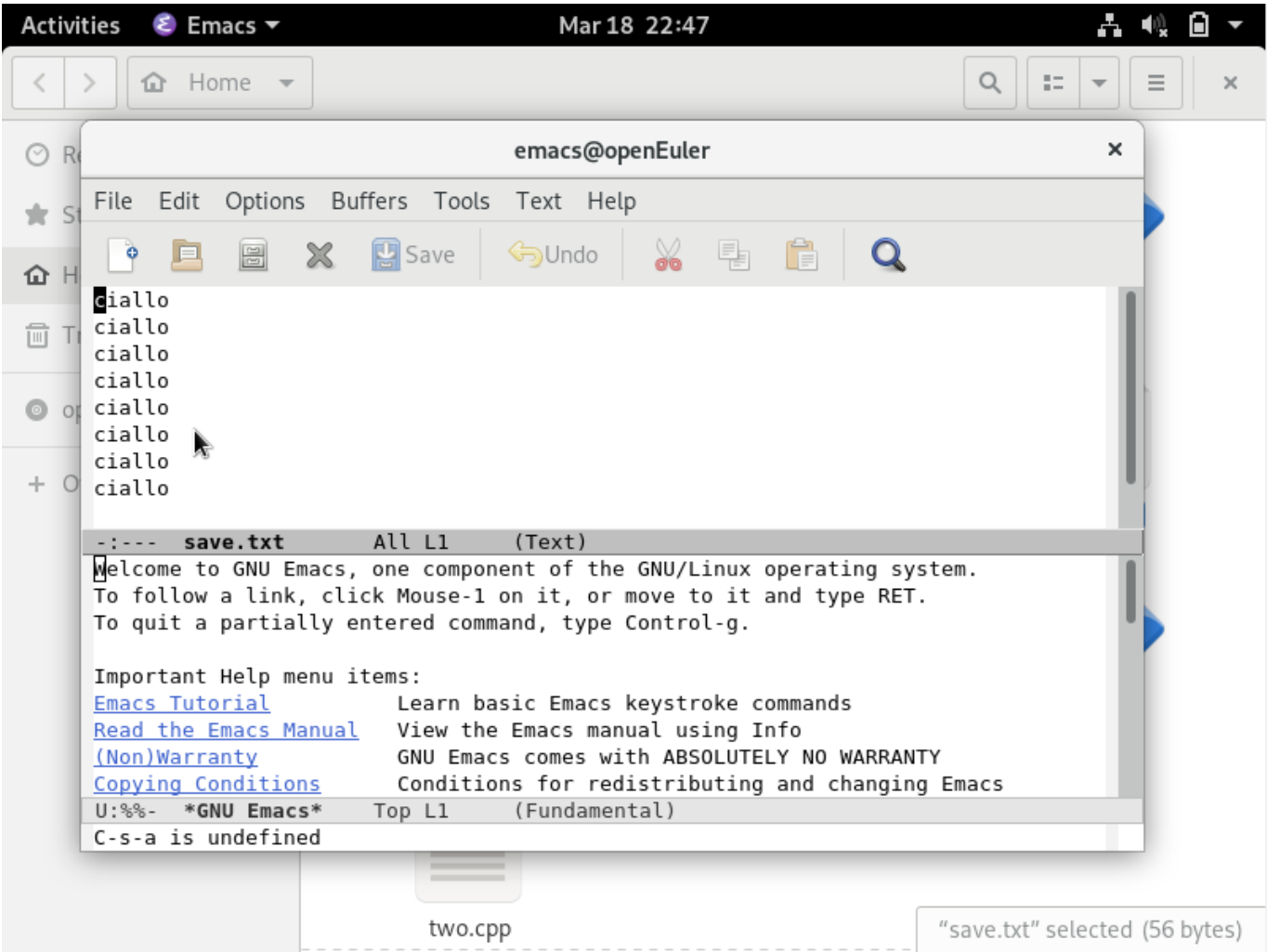


创建一个用来保存结果的文件save.txt

touch save.txt 编译代码并将运行的结果导入到save.txt

g++ san.cpp -o san ./san > save.txt

运行结果:



这表明：每次调用 `fork()` 后，当前进程都会复制出一个新的进程。

首次 `fork()` 后：2 个进程 再次 `fork()` 后：每个进程又复制出一个子进程，总数达到 4 个 第三次 `fork()` 后：总数达到 $(2^3 = 8)$ 个进程 因此，程序共输出 8 次 `ciao`，验证了进程复制的倍增效果。

进程独立性实验

代码:

The screenshot shows the Emacs editor interface. At the top, a system message box states: "Automatic suspend Computer will suspend very soon because of inactivity." The Emacs window title is "emacs@openEuler". The menu bar includes "File", "Edit", "Options", "Buffers", "Tools", "C++", and "Help". The toolbar contains icons for file operations and editing. The main text area displays a C++ program:

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

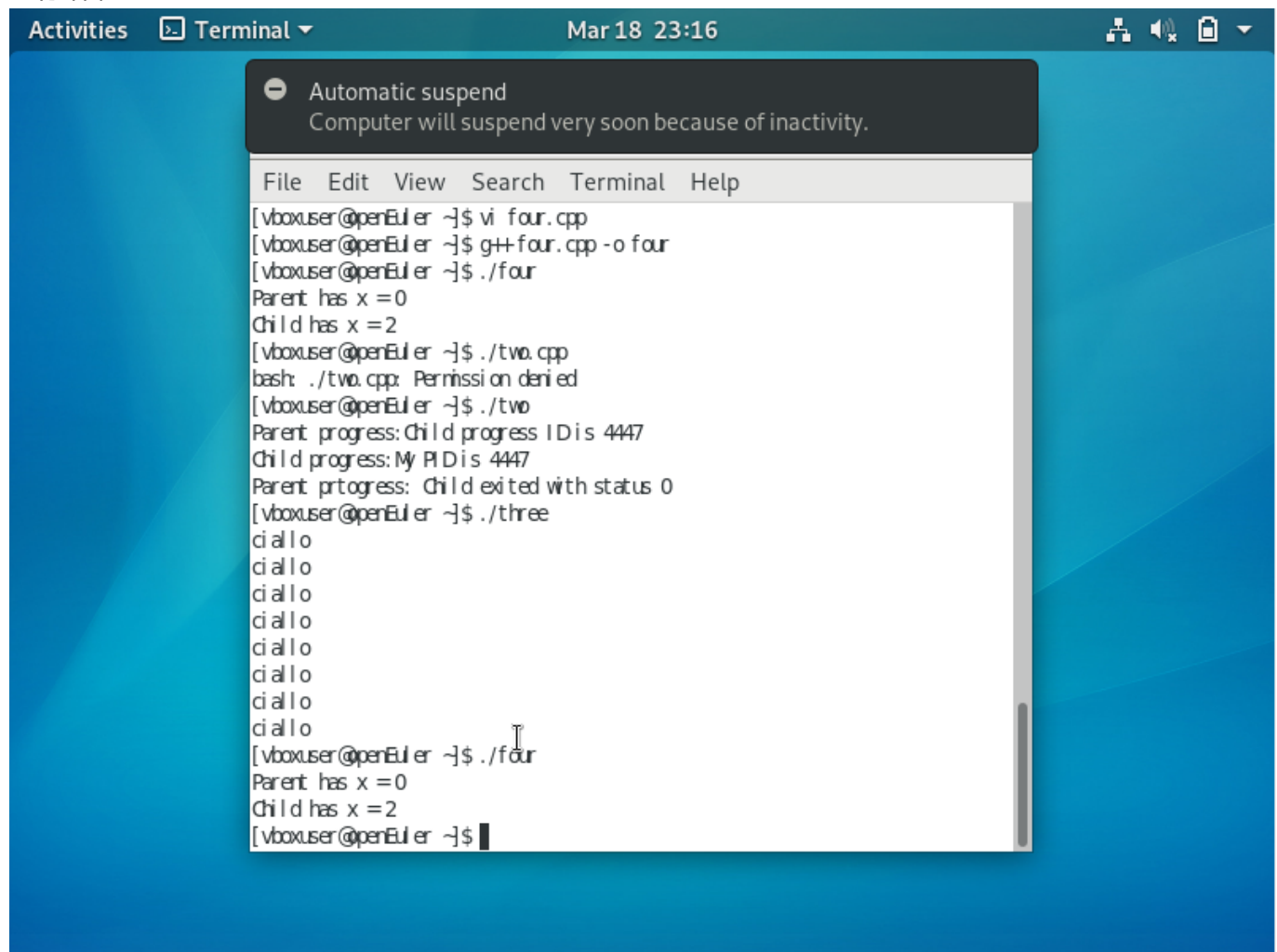
int main(){
    int x = 1;
    pid_t p = fork();
    if(p<0){
        perror("fork fail");
        exit(1);}
    else if(p==0)
        printf("Child has x = %d\n",++x);
    else
        printf("Parent has x = %d\n",--x);
}
```

Below the code, the Emacs status bar shows: "-:--- four.cpp Top L1 (C++//l Abbrev)". A tooltip or buffer window is visible, displaying the following text:

```
Welcome to GNU Emacs, one component of the GNU/Linux operating system.
To follow a link, click Mouse-1 on it, or move to it and type RET.
To quit a partially entered command, type Control-g.
U:%%- *GNU Emacs* Top L1 (Fundamental)
Mark set
```

At the bottom of the Emacs window, a buffer list shows "three.cpp", "two", and "four.cpp" selected (275 bytes).

运行结果:



```
Activities Terminal Mar 18 23:16
Automatic suspend
Computer will suspend very soon because of inactivity.
File Edit View Search Terminal Help
[vboxuser@openEuler ~]$ vi four.cpp
[vboxuser@openEuler ~]$ g++ four.cpp -o four
[vboxuser@openEuler ~]$ ./four
Parent has x = 0
Child has x = 2
[vboxuser@openEuler ~]$ ./two.cpp
bash: ./two.cpp: Permission denied
[vboxuser@openEuler ~]$ ./two
Parent progress: Child progress ID is 4447
Child progress: My PID is 4447
Parent progress: Child exited with status 0
[vboxuser@openEuler ~]$ ./three
ciao
ciao
ciao
ciao
ciao
ciao
ciao
ciao
ciao
[vboxuser@openEuler ~]$ ./four
Parent has x = 0
Child has x = 2
[vboxuser@openEuler ~]$
```

结果表明，父子进程在 `fork()` 调用后拥有各自独立的内存空间。

父进程对变量 `x` 执行自减操作，输出结果为0；子进程对变量 `x` 执行自增操作，输出结果为2。这验证了进程间变量互不干扰的特性。