

# Linux环境下的C语言代码实践

## 安装环境

启动虚拟机，以管理员身份登入后，依次键入如下代码：**sudo dnf install gcc**

```
sudo: apt: command not found
[tuotet17@vbox ~]$ sudo dnf install gcc
OS                               2.4 MB/s | 2.9 MB  00:01
Everything                       6.6 MB/s | 16 MB  00:02
EPOL                               2.4 MB/s | 5.7 MB  00:02
debuginfo                       4.1 MB/s | 4.4 MB  00:01
source                          2.3 MB/s | 1.7 MB  00:00
update                           5.6 MB/s | 9.2 MB  00:01
update-source                    463 kB/s | 321 kB  00:00
Dependencies resolved.
=====
Package      Architecture      Version                               Repository      Size
=====
Installing:
gcc           x86_64             12.3.1-62.oe2403sp1                 OS              34 M
Installing dependencies:
binutils     x86_64             2.41-14.oe2403sp1                   update          5.7 M
cpp          x86_64             12.3.1-62.oe2403sp1                 OS              11 M
gc           x86_64             0.2.4-1.oe2403sp1                   OS              250 k
glibc-devel x86_64             2.30-42.oe2403sp1                   OS              2.0 M
guile        x86_64             5.2.2-7-6.oe2403sp1                 OS              6.3 M
kernel-headers x86_64            6.6.0-02.0.0.06.oe2403sp1           update          1.8 M
libmpc       x86_64             1.3.1-1.oe2403sp1                   OS              64 k
libtool-ltdl x86_64             2.4.7-3.oe2403sp1                   OS              30 k
libxcrypt-devel x86_64            4.4.36-2.oe2403sp1                   OS             105 k
make         x86_64             1:4.4.1-2.oe2403sp1                   update          357 k
=====
Transaction Summary
=====
Install 11 Packages

Total download size: 62 M
Installed size: 221 M
Is this ok [y/N]: Operation aborted.
```

按y确认，安装好gcc编译器。

再键入**sudo dnf install vim**,重复如上操作，安装vim。

```
gpm-libs           x86_64             1.20.7-27.oe2403sp1                 OS              16 k
vim-common         x86_64             2:9.0.2092-17.oe2403sp1             update          7.0 M
vim-filessystem     noarch             2:9.0.2092-17.oe2403sp1             update          9.2 k
=====
Transaction Summary
=====
Install 4 Packages

Total download size: 9.4 M
Installed size: 39 M
Is this ok [y/N]: y
Downloading Packages:
(1/4): vim-enhanced-9.0.2092-17.oe2403sp1.x86_64.rpm           2.2 MB/s | 1.6 MB  00:00
(2/4): vim-filessystem-9.0.2092-17.oe2403sp1.noarch.rpm        277 kB/s | 9.2 kB  00:00
(3/4): gpm-libs-1.20.7-27.oe2403sp1.x86_64.rpm               12 kB/s | 16 kB   00:01
(4/4): vim-common-9.0.2092-17.oe2403sp1.x86_64.rpm           5.6 MB/s | 7.8 MB  00:01
-----
Total                                                                4.0 MB/s | 9.4 MB  00:02
retrieving repo key for OS unencrypted from http://repo.openeuler.org/openEuler-24.03-LTS-SP1/OS/x86_64/RPM-GPG-KEY-openeuler
OS                                                                15 kB/s | 3.0 kB  00:00
Importing GPG key 0x06756000:
  Userid      : "openeuler <openeuler@compass-ci.com>"
  Fingerprint: 0AA1 6BF9 F2CA 5244 010D CA96 3B47 7C60 B675 6000
  From        : http://repo.openeuler.org/openEuler-24.03-LTS-SP1/OS/x86_64/RPM-GPG-KEY-openeuler
Is this ok [y/N]: y
Key imported successfully
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction:
  Preparing                : vim-filessystem-2:9.0.2092-17.oe2403sp1.noarch                  1/1
  Installing               : vim-filessystem-2:9.0.2092-17.oe2403sp1.noarch                  1/4
  Installing               : vim-common-2:9.0.2092-17.oe2403sp1.x86_64                    2/4
( [ 060.050197] T0770) capability: warning: 'dnf' uses 32-bit capabilities (legacy support in use)
  Installing               : gpm-libs-1.20.7-27.oe2403sp1.x86_64                        3/4
  Installing               : vim-enhanced-2:9.0.2092-17.oe2403sp1.x86_64                  4/4
Running scriptlet: vim-enhanced-2:9.0.2092-17.oe2403sp1.x86_64                        4/4
Running scriptlet: vim-common-2:9.0.2092-17.oe2403sp1.x86_64                        4/4
Verifying                  : gpm-libs-1.20.7-27.oe2403sp1.x86_64                      1/4
Verifying                  : vim-common-2:9.0.2092-17.oe2403sp1.x86_64                  2/4
Verifying                  : vim-enhanced-2:9.0.2092-17.oe2403sp1.x86_64                3/4
Verifying                  : vim-filessystem-2:9.0.2092-17.oe2403sp1.noarch              4/4

Installed:
gpm-libs-1.20.7-27.oe2403sp1.x86_64          vim-common-2:9.0.2092-17.oe2403sp1.x86_64          vim-enhanced-2:9.0.2092-17.oe2403sp1.x86_64
vim-filessystem-2:9.0.2092-17.oe2403sp1.noarch

Complete!
[tuotet17@vbox ~]$
```

## 获取进程的PID

## 编写程序

输入指令 **vim pid.c**，使用vim开始编写c程序，点击输入代码，完成后点击esc，输入 **:wq**保存。



```
openeuler1234 [正在运行] - Oracle VirtualBox
管理 控制 视图 热键 设备 帮助

#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;
}
```

## 编译程序

输入指令 **gsc pid.c -o pid**,将名为pid.c的源文件编译成名为pid的可执行程序。

## 运行程序

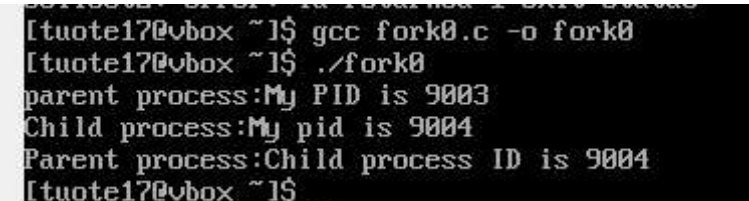
输入指令 **./pid**,运行程序，得到如图所示结果。



```
"pid.c" 11L, 165B written
[tuote17@vbox ~]$ gcc pid.c -o pid
[tuote17@vbox ~]$ ./pid
My process id is:8941\n[tuote17@vbox ~]$
```

## fork()示例0

编写与编译过程和上一节相同，本节和后续节不再赘述。输入指令 **./fork0**,运行程序，得到如图所示结果。

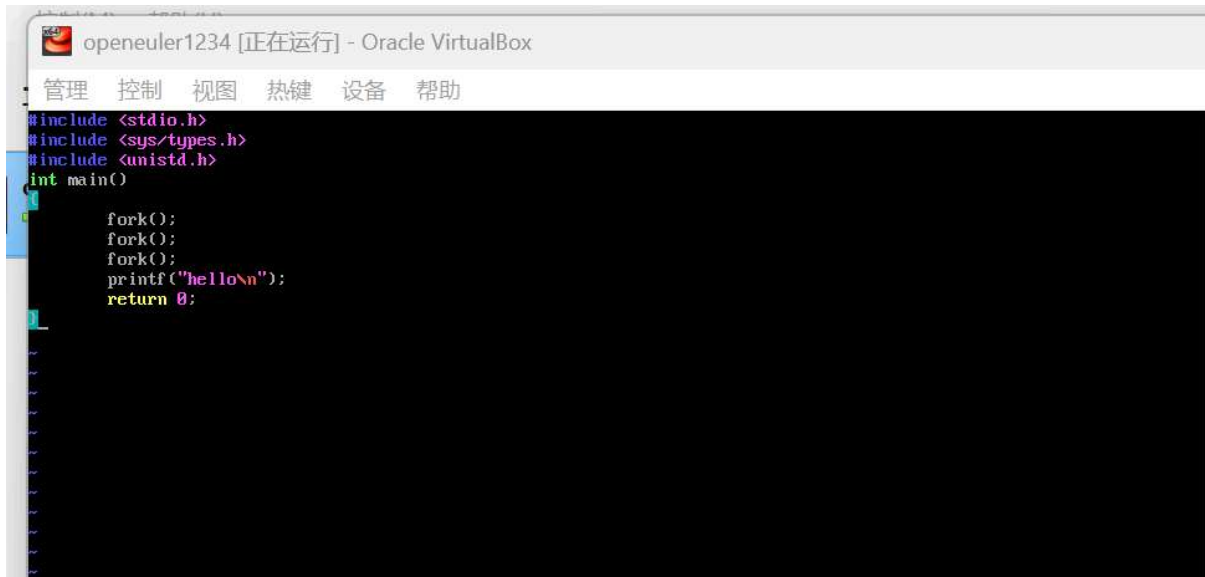


```
[tuote17@vbox ~]$ gcc fork0.c -o fork0
[tuote17@vbox ~]$ ./fork0
parent process:My PID is 9003
Child process:My pid is 9004
Parent process:Child process ID is 9004
[tuote17@vbox ~]$
```

fork()之后，进程9003和9004**交替运行**，导致先输出父进程的第一个printf，然后子进程的，然后父进程的第二个。

## fork()示例1

---



```
openeuler1234 [正在运行] - Oracle VirtualBox
管理 控制 视图 热键 设备 帮助
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
    fork();
    fork();
    fork();
    printf("hello\n");
    return 0;
}
```

输入指令 `./hello`,运行程序,得到如图所示结果。



```
Complete!
[tuote17@vbox ~]$ gcc hello.c -o hello
[tuote17@vbox ~]$ ./hello
hello
hello
hello
hello
hello
hello
hello
hello
[tuote17@vbox ~]$
```

由于3个fork(),程序变为了1->2->4->8个进程,最后输出了8条语句。

## fork()示例2

---

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<unistd.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);
    return 0;
}
```

输入指令 `./fork2`,运行程序,得到如图所示结果。

```
"fork2.c" [New] 20L, 279B written
[tuote17@vbox ~]$ gcc fork2.c -o fork2
[tuote17@vbox ~]$ ./fork2
Parent has x = 0
Child has x = 2
```

程序fork过后,父进程和子进程中的x为两个变量,所以分别输出0和2.