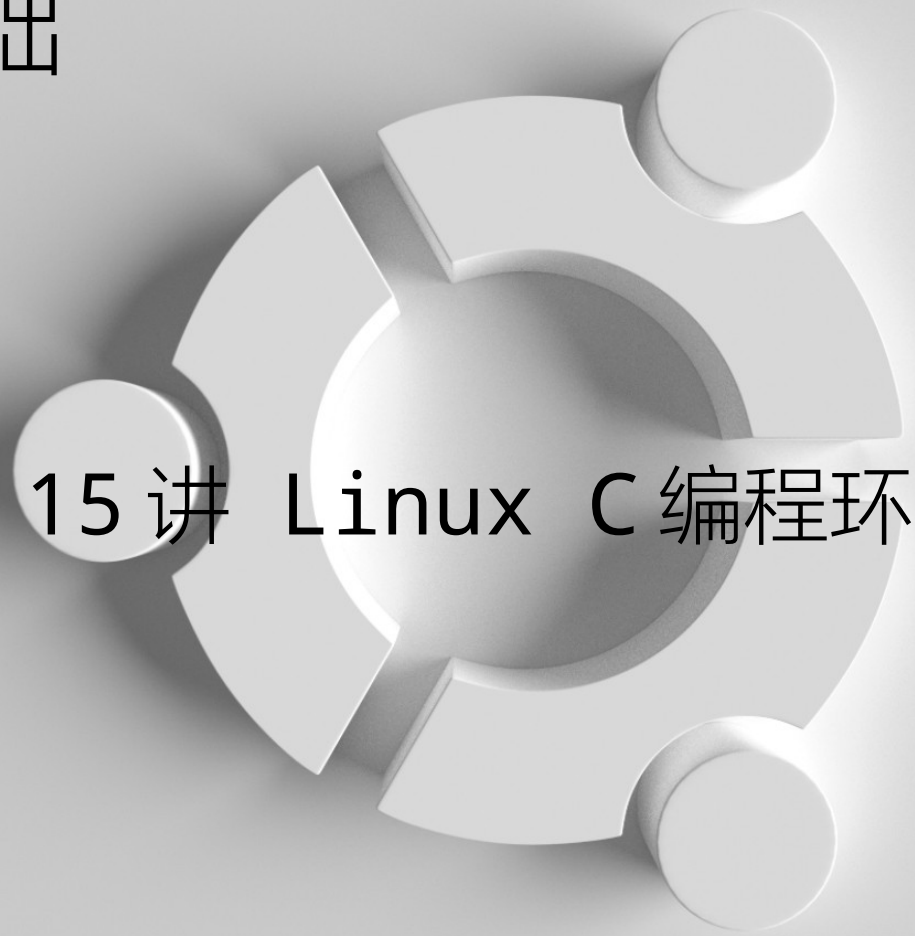
Linux 基础



第 15 讲 Linux C 编程环境

搭建 C 开发环境

- 如果你还没有搭建 C 环境，则需要安装 gcc：

  ```
  sudo apt install gcc
  ```

- gcc 是 GNU 的编译器集合。可以编译的语言有很多，但是主要用来编译 C/C++ 。

在 Unix （主要是 BSD 系列）上，因为开源协议限制，默认已经换成了 clang 。但是用户自己是可以安装 gcc 的。

选择合适的开发工具

- GUI 环境的 IDE 工具通常都比较笨重，自动补全太过智能有时候也会出问题，比如 vscode 总是自作聪明的在文件开头给你加上不需要的 import 操作导致问题，或者给你自动补全你不需要的量，并且还替换掉当前的变量。
- 通常来说，开发相对底层的工作，使用 vim 、 emacs 这些工具更好，当然使用 vim 开发 web 这种组件复杂的工作也是有很多扩展可以选择的，而专门的 web 开发工具相对来说效率更高。
- 尽管我们都会使用到 IDE ，但是我并不推荐过度的依赖这种重型工具。

选择合适的开发工具

- 在这里我们使用 `vim` 作为开发工具。
- 而 `gcc` 是用来编译的工具。
- 这可以更好的理解开发编译过程的工作方式。

编写 C 程序：程序的参数

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {

    for(int i=0; i < argc; i++) {
        printf("%s\n", argv[i]);
    }

    return 0;
}
```

- 根据 argc 输出每个参数，参数是通过 argv 传递的。
- argc 是参数的个数，最小为 1 ，因为 argv[0] 永远都是程序的名称。

编译你的程序

- 如果你写的文件是 first.c，那么使用 gcc 编译：

  ```
  gcc first.c
  ```

- 这会在当前目录生成 a.out 文件，而且任何文件都会编译成 a.out。因为没有指定文件名。一般在编译时，都要指定输出的文件名称，这可以使用 -o 选项完成：

  ```
  gcc -o first first.c
  ```

程序的返回值

- 程序最后的返回值为 0 表示正确。
- 非 0 值表示程序出错。
- 可以在 `shell` 中使用 `if-else` 关键字验证。

验证返回值的脚本

```
1  #!/bin/bash
2
3  if ./bin/outr -r abcdef ; then
4      echo '[OK]'
5  else
6      echo '[NOT OK]'
7  fi
8
9  echo ''
10
11 if ./bin/outr -r 12345 -r ; then
12     echo '[OK]'
13 else
14     echo '[NOT OK]'
15 fi
```

- 验证脚本和编译程序在同一目录。
- 在此目录有 `bin` 目录，编译好的程序都放在 `bin` 。

Linux 系统调用入门

- Linux 系统内核要和硬件设备通信，统一协调和管理硬件资源。并提供了 API 给程序调用。
- 在此基础上， glibc 又进行了一层封装，让内核接口更易于使用。
- 因为 Linux 内核除了一部分汇编部分代码，其他全部用 C 语言完成，其系统调用和 C 集成非常好，在 Linux 上调用系统接口是一种顺理成章的轻松事情。

系统联机文档

- 在 Linux/Unix 上，通过 man 可以直接查看库函数，系统调用的参考手册。
- 第 2 章节是系统调用参考手册。第 3 章节是库函数手册。
- 使用 man -k [关键词] 可以搜索相关文档。
- 使用 man syscalls 可以查看汇总的接口信息。

获取自己的 PID

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[]) {
    printf("%d\n", getpid());
    return 0;
}
```

getpid 调用可以获取进程自己的 PID 。

获取父进程的 PID

- 类似的，使用 getppid 可以获取父进程 PID 。
- 同样的， getppid 没有参数。
- 当进程创建子进程时，就会标记子进程的父进程是谁。
- init 进程（ systemd 进程， PID=1 ）是进程树的顶层。

创建子进程

- 进程控制是一个比较麻烦的工作，要管理进程就要先创建。这里先了解如何创建子进程。
- Linux/Unix 提供了系统调用 fork 用于创建子进程。
- fork 没有参数。

理解 fork

- fork 的返回值在父进程和子进程中不同，父进程中返回子进程的 PID，子进程中返回 0 。
- 父进程和子进程都继续执行 fork 之后的代码。
- 如果失败， fork 调用返回 -1 。

理解 fork

- 根据 fork 返回值的不同，可以控制父进程和子进程执行不同的代码。
- 一开接触 fork 感觉难以理解，并且难以控制程序的逻辑，但是只要知道，系统是从当前进程复制了一份出来，并让两个进程继续执行下面的代码，编码就相对清晰很多。因为我们编写的代码，是要给两个进程执行的。