Linux 基础

第 15 讲　Linux C 编程基础

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
```

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。

通过参数控制程序行为

- 实现简单的 `echo`，并通过 `-r` 控制可以反转输出字符串。
- 示例代码参考源代码文件。

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。

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。

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。

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。