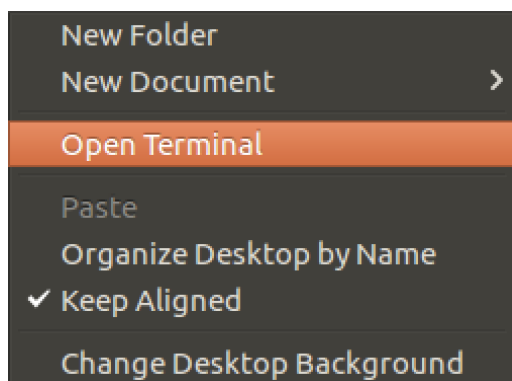


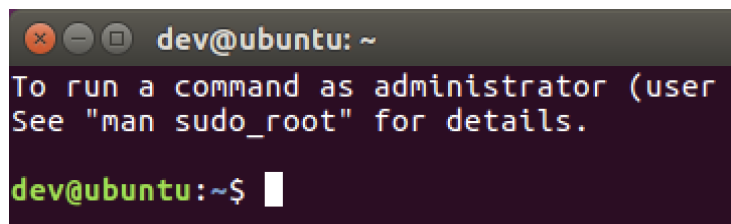
第二章 Linux 常用命令简介

在编译的过程中，会使用到一些 Linux 命令或工具，本章将介绍使用到的命令或工具，包括功能、参数、以及一些外部工具的安装等。

以下命令的测试需要打开终端 (Terminal)，可以在 Ubuntu 桌面上右键，点击 Open Terminal 来打开。



执行后打开的窗体如下：



当前显示的窗体,包含以下信息：

- 1：当前登录的用户名为 dev。
- 2：当前的主机名为 ubuntu 。
- 3：当前用户所在目录为 ~ ，一般也称为 Home 或者家目录, 普通用户 dev 的~绝对路径一般为/home/dev。
- 4：当前用户为普通用户，显示为\$，如果显示为#，一般表示为 root 用户，root 用户为 linux 的超级管理员。

如果需要切换到管理员身份进行执行命令，则需要在执行的命令行前增加 sudo。

比如使用管理员权限查看/tmp 目录存在的文件列表，在终端输入 **sudo ls /tmp** ，终端则需要判断权限，提示输入密码，密码输入过程是不回显的。

```
dev@ubuntu: ~  
dev@ubuntu:~$ sudo ls /tmp  
[sudo] password for dev:  
config-err-zp09YP  
systemd-private-b5cb6c97d41f4a5b88fcadc6055a4262-colord.service-GC2vk0  
systemd-private-b5cb6c97d41f4a5b88fcadc6055a4262-rtkit-daemon.service-LNTL5L  
systemd-private-b5cb6c97d41f4a5b88fcadc6055a4262-systemd-timesyncd.service-kAPhN5  
5  
unity_support_test.0  
VMwareDnD  
vmware-root  
dev@ubuntu:~$
```

一 常用命令

1. ls

查看当前目录，ls 空格，查看/tmp 目录，ls /tmp

```
dev@ubuntu: ~  
dev@ubuntu:~$ ls  
Desktop      Downloads      Music          Public         Videos  
Documents    examples.desktop Pictures        Templates  
dev@ubuntu:~$ ls /tmp  
config-err-zp09YP  
systemd-private-b5cb6c97d41f4a5b88fcadc6055a4262-colord.service-GC2vk0  
systemd-private-b5cb6c97d41f4a5b88fcadc6055a4262-rtkit-daemon.service-LNTL5L  
systemd-private-b5cb6c97d41f4a5b88fcadc6055a4262-systemd-timesyncd.service-kAPhN5  
5  
unity_support_test.0  
VMwareDnD  
vmware-root  
dev@ubuntu:~$
```

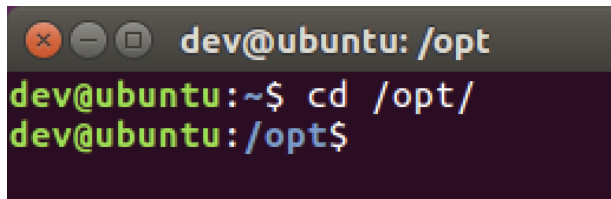
如果需要调整排列，可以输入 ls -l，一般的 linux 提供了 ll 命令。

比如：ls -l /tmp 等同于 ll /tmp。

```
dev@ubuntu: ~  
dev@ubuntu:~$ ls -l /tmp/  
total 24  
-rw----- 1 dev dev 0 Oct 28 18:41 config-err-zp09YP  
drwx----- 3 root root 4096 Oct 28 18:38 systemd-private-b5cb6c97d41f4a5b88fcadc6055a4262-colord.service-GC2vk0  
drwx----- 3 root root 4096 Oct 28 18:38 systemd-private-b5cb6c97d41f4a5b88fcadc6055a4262-rtkit-daemon.service-LNTL5L  
drwx----- 3 root root 4096 Oct 28 18:37 systemd-private-b5cb6c97d41f4a5b88fcadc6055a4262-systemd-timesyncd.service-kAPhN5  
-rw-rw-r-- 1 dev dev 6 Oct 28 19:15 test.c  
-rw-rw-r-- 1 dev dev 0 Oct 28 18:41 unity_support_test.0  
drwxrwxrwt 2 root root 4096 Oct 28 18:38 VMwareDnD  
drwx----- 2 root root 4096 Oct 28 18:38 vmware-root  
dev@ubuntu:~$
```

2. cd

进入在某个目录，比如进入到 /opt 目录。



```
dev@ubuntu: /opt
dev@ubuntu:~$ cd /opt/
dev@ubuntu:/opt$
```

3. cp

复制文件或文件夹到目标路径

cp 文件 目标路径

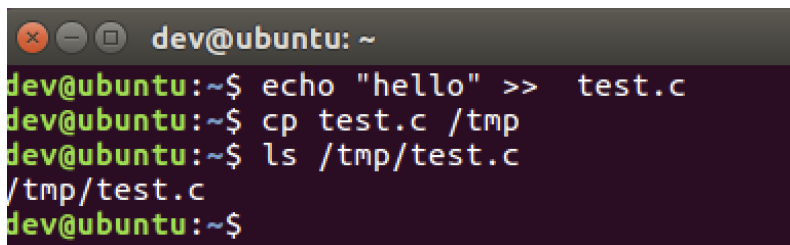
cp -rf 文件夹 目录路径

注意

参数 r 表示递归文件夹里面所有文件夹

参数 f 表示如果目标文件无法打开，则尝试先删除再复制。

创建 test.c 文件，里面填充 hello 内容，然后将 test.c 文件复制到/tmp 目录。



```
dev@ubuntu: ~
dev@ubuntu:~$ echo "hello" >> test.c
dev@ubuntu:~$ cp test.c /tmp
dev@ubuntu:~$ ls /tmp/test.c
/tmp/test.c
dev@ubuntu:~$
```

如果目标文件有可能存在，想再次覆盖的话，并且不想提示，可以在 cp 前面增加反斜杠 \。

\cp test.c /tmp/test.c

4. ps

查看当前运行的进程

一般使用 ps 的 -e 和 -f 参数，比如执行 **ps -ef** 。

```
dev@ubuntu: ~  
dev@ubuntu:~$ ps -ef  
UID          PID    PPID  C  STIME TTY          TIME CMD  
root           1         0  0   18:37 ?           00:00:01 /sbin/init auto noprompt  
root           2         0  0   18:37 ?           00:00:00 [kthreadd]  
root           3         2  0   18:37 ?           00:00:00 [ksoftirqd/0]  
root           5         2  0   18:37 ?           00:00:00 [kworker/0:0H]  
root           7         2  0   18:37 ?           00:00:00 [rcu_sched]  
root           8         2  0   18:37 ?           00:00:00 [rcu_bh]  
root           9         2  0   18:37 ?           00:00:00 [migration/0]  
root          10         2  0   18:37 ?           00:00:00 [watchdog/0]  
root          11         2  0   18:37 ?           00:00:00 [watchdog/1]  
root          12         2  0   18:37 ?           00:00:00 [migration/1]  
root          13         2  0   18:37 ?           00:00:00 [ksoftirqd/1]
```

返回结果简介：

1：PID 表示进程的 ID。

2：PPID 表示进程的父进程 ID。

3：CMD 表示启动时候执行的命令参数或是系统进程。

5. grep

一般用来过滤结果,或者从文件中查找内容

比如：从 test.c 文件中搜索 he 关键字，则执行 **grep he test.c**

从当前目录所有文件中搜索 hello 关键字，则执行 **grep hello ./* -r**

```
dev@ubuntu: ~  
dev@ubuntu:~$ man ps  
dev@ubuntu:~$ grep he test.c  
hello  
dev@ubuntu:~$ grep hello ./* -r  
./test.c:hello  
dev@ubuntu:~$
```

6. |

一般称为管道(竖线)。将执行命令的结果作为其他命令的参数。

比如：在当前运行的进程列表进行过滤，只是显示 thread 为关键字的。

ps -ef | grep thread

```
dev@ubuntu: ~  
dev@ubuntu:~$ ps -ef | grep thread  
root      2      0  0 18:37 ?        00:00:00 [kthreadd]  
dev       5380   5274  0 19:29 pts/1    00:00:00 grep --color=auto thread  
dev@ubuntu:~$
```

7. find

一般多用于来查找文件

比如：查找当前目录，文件名包含 tes 的文件。

find ./ -name *tes*

```
dev@ubuntu: ~  
dev@ubuntu:~$ find ./ -name \*tes\  
./test.c  
./Templates  
dev@ubuntu:~$
```

8. ldd

一般用于来分析某个可执行文件或动态库依赖的其他动态库，在后面章节对编译的 QT 可执行文件分析很有帮助。

比如：分析 libcrypt.so 依赖哪些动态库文件

ldd /usr/lib/x86_64-linux-gnu/libcrypt.so

```
dev@ubuntu: ~  
dev@ubuntu:~$ ldd /usr/lib/x86_64-linux-gnu/libcrypt.so  
linux-vdso.so.1 => (0x00007fff6eb90000)  
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f8a3a6b6000)  
/lib64/ld-linux-x86-64.so.2 (0x0000559cacb2d000)  
dev@ubuntu:~$
```

9. nm

查看库导出的函数，比如查看 libutil.a 静态库导出了哪些函数。

nm /usr/lib/x86_64-linux-gnu/libutil.a

```
dev@ubuntu: ~  
dev@ubuntu:~$ nm /usr/lib/x86_64-linux-gnu/libutil.a  
login.o:  
                 U basename  
                 U endutent  
                 U errno  
                 U free  
                 U getpid  
                 U _GLOBAL_OFFSET_TABLE_  
0000000000000000 T login  
                 U malloc  
                 U memchr  
                 U pututline
```

10.file

查看文件类型，比如分析动态库或者.data 文件具体是什么格式，在 Linux 系统很多情况不是根据文件后缀来区分文件类型的。

比如新增一个文本文件为 so 后缀

```
echo "hello linux" >> test-file.so
```

```
file test-file.so
```

```
dev@ubuntu: ~  
dev@ubuntu:~$ echo "hello linux" >> test-file.so  
dev@ubuntu:~$ file test-file.so  
test-file.so: ASCII text  
dev@ubuntu:~$
```

最后的显示 test-file.so 的文件类型为 ASCII text，表示为文本文件。

11.chmod

一般多用于给某个文件增加某个权限，在 Linux 系统，即使是可执行文件，双击或者直接执行的前提是需要可执行权限，称为 x。

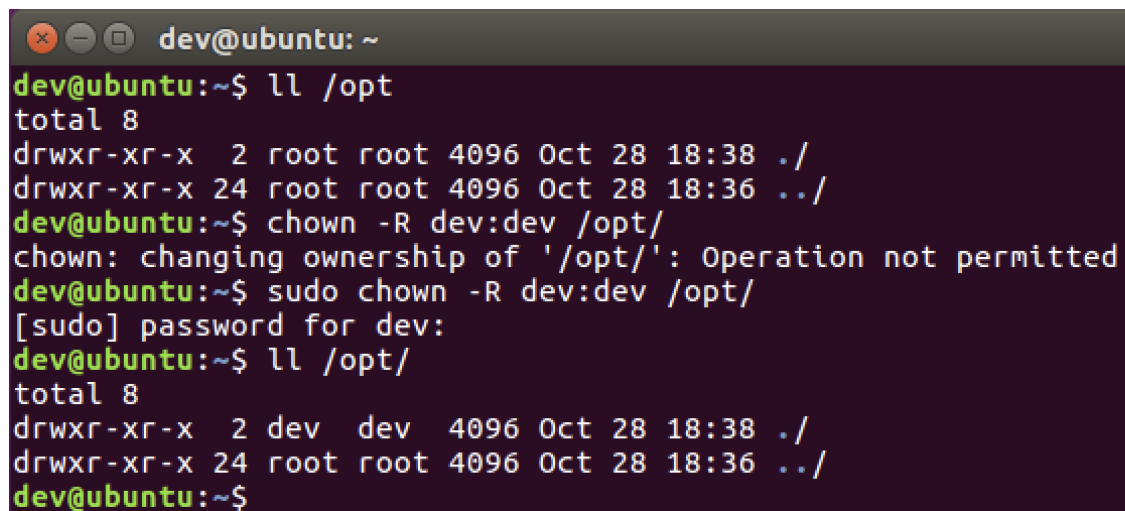
```
dev@ubuntu: ~  
dev@ubuntu:~$ ll test-file.so  
-rw-rw-r-- 1 dev dev 12 Oct 28 19:40 test-file.so  
dev@ubuntu:~$ chmod +x test-file.so  
dev@ubuntu:~$ ll test-file.so  
-rwxrwxr-x 1 dev dev 12 Oct 28 19:40 test-file.so*  
dev@ubuntu:~$
```

注意在 Linux 系统，可以看到左侧的从-rw-rw-r--变更为-rwxrwxr-x，里面增加了 x，表示可执行权限。

12.chown

调整文件或文件夹的权限,比如/opt 目录本身是属于 root 用户，一般用户默认没有权限写，

将/opt 的目录以及里面所有的文件所有者更改为 dev 用户。



```
dev@ubuntu: ~  
dev@ubuntu:~$ ll /opt  
total 8  
drwxr-xr-x  2 root root 4096 Oct 28 18:38 ./  
drwxr-xr-x 24 root root 4096 Oct 28 18:36 ../  
dev@ubuntu:~$ chown -R dev:dev /opt/  
chown: changing ownership of '/opt/': Operation not permitted  
dev@ubuntu:~$ sudo chown -R dev:dev /opt/  
[sudo] password for dev:  
dev@ubuntu:~$ ll /opt/  
total 8  
drwxr-xr-x  2 dev  dev  4096 Oct 28 18:38 ./  
drwxr-xr-x 24 root root 4096 Oct 28 18:36 ../  
dev@ubuntu:~$
```

现在/opt 用户显示为 dev 用户所拥有，

./ 表示当前目录，/opt 里面的./则表示 /opt 目录。

../ 表示上一层目录，/opt 里面的../则表示 /opt 上层目录，也就是 / 目录，我们称为根目录，仍然为 root 用户所拥有。

13.其他命令

针对其他命令，可以在终端输入 man 命令行，表示查看当前命令行的帮助。

比如：查看 ps 的参数说明，则可以输入 man ps，通过按空格进行翻页，按 q 表示退出。

```
dev@ubuntu: ~
PS(1)                                User Commands                                PS(1)
NAME
    ps - report a snapshot of the current processes.
SYNOPSIS
    ps [options]
DESCRIPTION
    ps displays information about a selection of the active processes.  If
    you want a repetitive update of the selection and the displayed
    information, use top(1) instead.

    This version of ps accepts several kinds of options:

    1  UNIX options, which may be grouped and must be preceded by a dash.
    2  BSD options, which may be grouped and must not be used with a dash.
    3  GNU long options, which are preceded by two dashes.

    Options of different types may be freely mixed, but conflicts can
    appear.  There are some synonymous options, which are functionally
    identical, due to the many standards and ps implementations that this
    ps is compatible with.
Manual page ps(1) line 1 (press h for help or q to quit)
```

大部分工具在 Linux 里面比较常见的启动参数是--help 表示查看使用命令的帮助。

比如 **ps --help**

```
dev@ubuntu: ~
dev@ubuntu:~$ ps --help
Usage:
  ps [options]

Try 'ps --help <simple|list|output|threads|misc|all>'
  or 'ps --help <s|l|o|t|m|a>'
for additional help text.

For more details see ps(1).
dev@ubuntu:~$
```

二 调试与分析工具

当开发完应用程序，发现启动很卡或者想分析一下某些应用调用了什么接口，strace 以及 ltrace 工具非常有帮助。

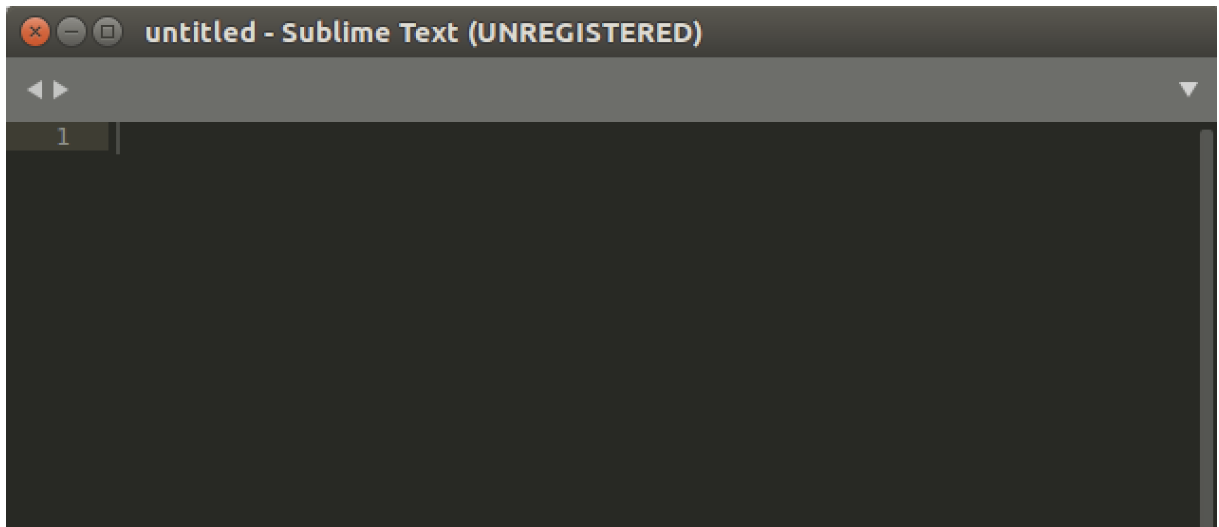
1. strace

2. ltrace

三 安装与更新其他工具

1. sublime text

需要去到官方 <http://www.sublimetext.com/3> 页面，下载 **64 bit .tar.bz2** 的包，然后双击已下载的 bz2 包，解压缩后直接运行里面的 sublime_text 可执行文件。



该工具用于后面浏览或编辑 QT 源码库的部分文件。

2. 安装 vim 与 git

vim 是文本编辑工具，git 是方便后面获取开源代码。

```
sudo apt install vim git -y
```

3. 安装 clang 12

注意不能直接安装默认的 clang 版本,会导致 QT 的源码编译失败。

增加 clang 的官方源步骤

1. 获取 key

```
wget -O - https://apt.llvm.org/llvm-snapshot.gpg.key | sudo apt-key add -
```

2. 增加源，编辑/etc/apt/sources.list 文件，增加以下内容

```
deb http://apt.llvm.org/xenial/ llvm-toolchain-xenial main
```

```
deb-src http://apt.llvm.org/xenial/ llvm-toolchain-xenial main
# 11
deb http://apt.llvm.org/xenial/ llvm-toolchain-xenial-11 main
deb-src http://apt.llvm.org/xenial/ llvm-toolchain-xenial-11 main
# 12
deb http://apt.llvm.org/xenial/ llvm-toolchain-xenial-12 main
deb-src http://apt.llvm.org/xenial/ llvm-toolchain-xenial-12 main
```

3. **sudo apt install clang-12 -y**

安装成功后，需要将 clang 的 bin 目录，设置到系统的 PATH 路径，方便在终端里面执行 clang 能找到正确的位置。编辑文件 `~/.bashrc`，在文件末尾新增一行内容。

```
export PATH=/usr/lib/llvm-12/bin:$PATH
```

使设置的环境变量生效，需要执行 **source ~/.bashrc**。验证设置是否正确，使用 **which clang**。

4. 安装 cmake

为了后面正常对 CMakeLists.txt 管理的项目进行配置并生成 Makefile 文件，需要安装 cmake，不建议使用 apt 的方式进行安装，因为 cmake 的版本比较旧，建议去 cmake 官方网站下载最新的二进制文件。

官方网站：<https://cmake.org/download/>，下载最新的版本，目前较新的是 cmake-3.22.0-rc2-linux-x86_64.tar.gz。

解压缩 **tar zxvf cmake-3.22.0-rc2-linux-x86_64.tar.gz /opt/tools/**，将 cmake 的可执行文件设置到系统的 PATH 变量中，编辑文件 `~/.bashrc`，在文件末尾新增一行内容

```
export PATH=/opt/tools/cmake-3.22.0-rc2-linux-x86_64/bin:$PATH
```

然后将设置的立刻生效，需要执行 **source ~/.bashrc**，验证 cmake 是否设置生效，执行 **which cmake** 是否返回刚才的路径。