

汇总一些实用的Linux小技巧

ZhengNL 嵌入式大杂烩 2023-02-01 21:50 发表于广东

大家好，我是ZhengN。本次给大家分享一些Linux的实用小技巧。

1、查看文件校验值

在文件进行拷贝或者进行传输的时候，可能有损坏或者被修改的可能，这时候可以查看校验值来确认一下。

比如我们平时工作需要用到其它组给我们提供的一些对接的程序，每次程序运行不符合他们的预期的时候，我们都会对一下两边的md5校验值。

生成文件的校验值的方法有很多种，常用的有md5sum校验、crc校验、sum校验等。

命令分别为：

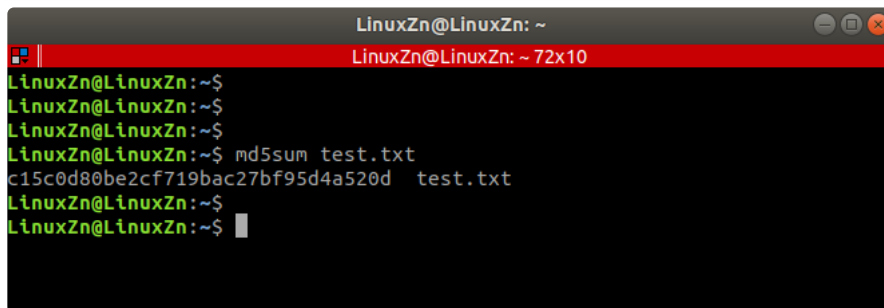
```
md5sum file_name
cksum file_name
sum 算法参数 file_name
```

例如：

我们以一个test.txt文件为例：

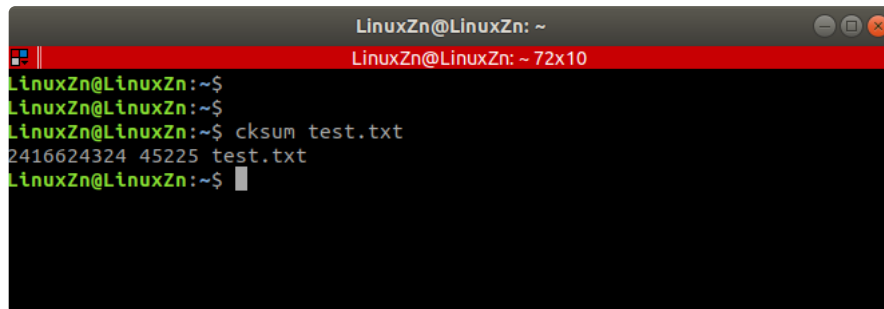
- md5sum校验

```
md5sum test.txt
```

A terminal window titled 'LinuxZn@LinuxZn: ~' with a red title bar. The window shows the command 'md5sum test.txt' being executed. The output is 'c15c0d80be2cf719bac27bf95d4a520d test.txt'. The prompt is 'LinuxZn@LinuxZn:~\$'.

- crc校验

```
cksum test.txt
```

A terminal window titled 'LinuxZn@LinuxZn: ~' with a red title bar. The window shows the command 'cksum test.txt' being executed. The output is '2416624324 45225 test.txt'. The prompt is 'LinuxZn@LinuxZn:~\$'.

- sum校验

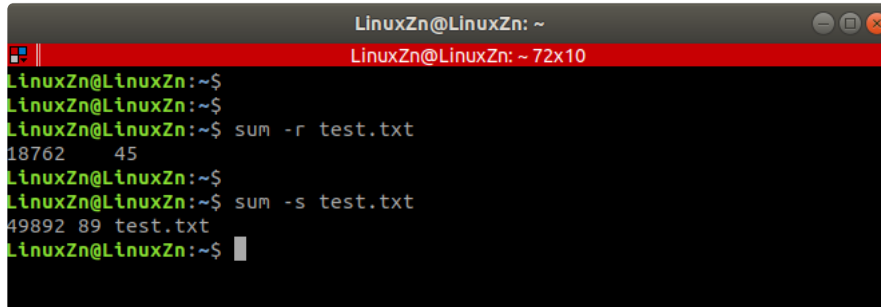
sum校验有两种算法，我们可以通过参数进行配置：

-r: 表示使用system v算法。

-s: 表示使用BSD算法。

我们不进行配置时，默认用的是system v算法。

```
sum -r test.txt
sum -s test.txt
```



```
LinuxZn@LinuxZn: ~
LinuxZn@LinuxZn: ~ 72x10
LinuxZn@LinuxZn:~$
LinuxZn@LinuxZn:~$
LinuxZn@LinuxZn:~$ sum -r test.txt
18762 45
LinuxZn@LinuxZn:~$
LinuxZn@LinuxZn:~$ sum -s test.txt
49892 89 test.txt
LinuxZn@LinuxZn:~$
```

2、查找文件位置

(1) locate

查找文件大家都习惯用find吧，但我觉得有时候locate更快一些，所以我一般都会先使用locate。

locate 与 find 不同: find 是去硬盘找，locate 只在 /var/lib/slocate 资料库中找。locate 的速度比 find 快，它并不是真的查找，而是查数据库。


有些系统可能不带有locate，需要自己安装。比如，Ubuntu可以输入如下命令进行安装：

```
apt-get update
apt-get install mlocate
```

locate查找文件的命令很简单：

```
locate file_name
```

比如：



```
LinuxZn@LinuxZn:~$ locate stdio.h
/home/LinuxZn/100ask_linuxsdk/Bulldroot_2019.02/package/ddrescue/0001-to.cc-add-stdio.h-include.patch
/home/LinuxZn/100ask_linuxsdk/Bulldroot_2019.02/package/stress-ng/0003-test-test-bsd-wchar-Explicitly-include-stdio.h.patch
/home/LinuxZn/100ask_linuxsdk/Bulldroot_2020.02.x/package/bash/0017-input.h-add-missing-include-on-stdio.h.patch
/home/LinuxZn/100ask_linuxsdk/Linux-4.9.88/arch/powerpc/boot/stdio.h
/home/LinuxZn/100ask_linuxsdk/Linux-4.9.88/arch/powerpc/xmon/nostdio.h
/home/LinuxZn/100ask_linuxsdk/ToolChain/gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf/arm-linux-gnueabihf/include/c++/6.2.1/tr1/stdio.h
/home/LinuxZn/100ask_linuxsdk/ToolChain/gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf/arm-linux-gnueabihf/libc/usr/include/stdio.h
/home/LinuxZn/100ask_linuxsdk/ToolChain/gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf/arm-linux-gnueabihf/libc/usr/include/bits/stdio.h
/home/LinuxZn/100ask_linuxsdk/ToolChain/gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf/lib/gcc/arm-linux-gnueabihf/6.2.1/include/ssp/stdio.h
/usr/include/stdio.h
/usr/include/c++/7/tr1/stdio.h
/usr/include/glib-2.0/glib/gstdio.h
/usr/include/unicode/ustdio.h
/usr/include/x86_64-linux-gnu/bits/stdio.h
/usr/lib/x86_64-linux-gnu/perl/5.26.1/CORE/nostdio.h
LinuxZn@LinuxZn:~$
```

(2) find

find命令可以用名字、类型、所属人、大小等来进行搜索。

搜索文件基本语法：

```
find path -option file_name
```

如使用名字来搜索stdio.h文件：

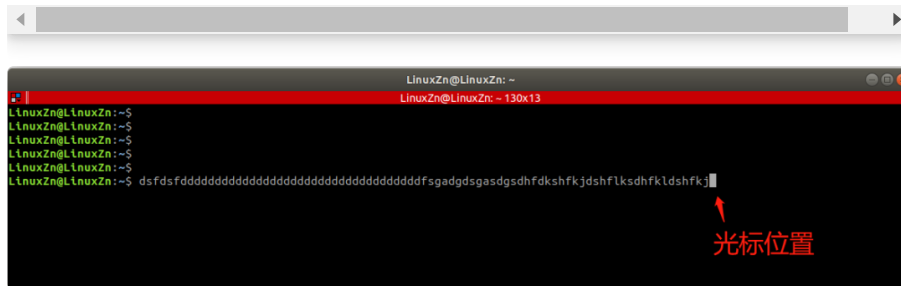
```
find / -name stdio.h
```

```
LinuxZn@LinuxZn:~$ sudo find / -name stdio.h
find: '/run/user/1001/gvfs': 权限不够
/home/LinuxZn/100ask_linuxull-sdk/uboot-2018.03/include/stdio.h
/home/LinuxZn/100ask_linuxull-sdk/Linux-4.9.88/arch/powerpc/boot/stdio.h
/home/LinuxZn/100ask_linuxull-sdk/ToolChain/gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf/arm-linux-gnueabihf/libc/usr/include/stdio.h
/home/LinuxZn/100ask_linuxull-sdk/ToolChain/gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf/arm-linux-gnueabihf/libc/usr/include/bits/stdio.h
/home/LinuxZn/100ask_linuxull-sdk/ToolChain/gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf/arm-linux-gnueabihf/include/c++/6.2.1/tr1/stdio.h
/home/LinuxZn/100ask_linuxull-sdk/ToolChain/gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf/lib/gcc/arm-linux-gnueabihf/6.2.1/include/ssp/stdio.h
/usr/include/x86_64-linux-gnu/bits/stdio.h
/usr/include/stdio.h
/usr/include/c++/7/tr1/stdio.h
LinuxZn@LinuxZn:~$
```

3、命令行编辑技巧

我们在终端里误输入了一些比较长的内容：

```
LinuxZn@LinuxZn:~$ dsfdsfdddddddddddddddddddddddddddfsgadgsgasgdsdhfdkshfkjdshfklshfklshfklshf
```



怎么比较快的删除掉呢？疯狂地按退格键当然可以达到目的。但是有更快速的方法：

输入快捷键 **ctrl+u** 即可把光标前面的内容全删掉。除此之外，还有如下几个实用且常用的快捷键：

- ctrl+k：把光标后面的内容全删掉。
- ctrl+a：光标移到开头处。
- ctrl+e：光标移动到末尾处。

除此之外，命令行还有很多实用常用、实用不常用的快捷方式，感兴趣的小伙伴可以自己去学习。

4、查看某个进程的pid

命令：

```
pidof process_name
```

如：

```
LinuxZn@LinuxZn:~$
LinuxZn@LinuxZn:~$
LinuxZn@LinuxZn:~$ pidof kcalc
5150
LinuxZn@LinuxZn:~$
```

5、查看某些进程的一些运行情况

top命令可以查看进程的一些信息，但是系统运行的进程过多，不利于我们查看某些进程的运行情况，如：

```
LinuxZn@LinuxZn: ~ 171x62
top - 01:32:23 up 2:38, 1 user, load average: 0.99, 0.62, 0.30
任务: 441 total, 2 running, 341 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.5 us, 0.3 sy, 0.0 ni, 99.1 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 8124336 total, 1999852 free, 3328812 used, 2795672 buff/cache
KiB Swap: 8787964 total, 8787964 free, 0 used, 4366244 avail Mem

进程 USER      PR  NI   VIRT   RES   SHR  %CPU %MEM    TIME+  COMMAND
5533 LinuxZn    20   0   4504    856   722 R   94.1  0.0   3:46.54 test_x86
4733 LinuxZn    20   0 36.439g 132856 54152 S   5.9  1.6   0:03.07 code
1 root        20   0 225632    954   6844 S   0.0  0.1   0:05.14 systemd
2 root        20   0      0      0      0 S   0.0  0.0   0:00.03 kthreadd
3 root        0 -20      0      0      0 I   0.0  0.0   0:00.00 rcu_gp
4 root        0 -20      0      0      0 I   0.0  0.0   0:00.00 rcu_par_gp
6 root        0 -20      0      0      0 I   0.0  0.0   0:00.00 kworker/0:0H
9 root        0 -20      0      0      0 I   0.0  0.0   0:00.00 mm_percpu_wq
10 root       20   0      0      0      0 S   0.0  0.0   0:00.04 ksoftirqd/0
11 root       20   0      0      0      0 I   0.0  0.0   0:04.16 rcu_sched
12 root       rt   0      0      0      0 S   0.0  0.0   0:00.05 migration/0
13 root      -51   0      0      0      0 S   0.0  0.0   0:00.00 idle_inject/0
14 root       20   0      0      0      0 S   0.0  0.0   0:00.00 cpuphp/0
15 root       20   0      0      0      0 S   0.0  0.0   0:00.00 cpuphp/1
16 root      -51   0      0      0      0 S   0.0  0.0   0:00.00 idle_inject/1
17 root       rt   0      0      0      0 S   0.0  0.0   0:00.76 migration/1
18 root       20   0      0      0      0 S   0.0  0.0   0:00.02 ksoftirqd/1
20 root       0 -20      0      0      0 I   0.0  0.0   0:00.00 kworker/1:0H-kb
21 root       20   0      0      0      0 S   0.0  0.0   0:00.00 cpuphp/2
22 root      -51   0      0      0      0 S   0.0  0.0   0:00.00 idle_inject/2
23 root       rt   0      0      0      0 S   0.0  0.0   0:00.75 migration/2
24 root       20   0      0      0      0 S   0.0  0.0   0:00.03 ksoftirqd/2
26 root       0 -20      0      0      0 I   0.0  0.0   0:00.00 kworker/2:0H
27 root       20   0      0      0      0 S   0.0  0.0   0:00.00 cpuphp/3
28 root      -51   0      0      0      0 S   0.0  0.0   0:00.00 idle_inject/3
29 root       rt   0      0      0      0 S   0.0  0.0   0:00.75 migration/3
30 root       20   0      0      0      0 S   0.0  0.0   0:00.02 ksoftirqd/3
32 root       0 -20      0      0      0 I   0.0  0.0   0:00.00 kworker/3:0H-kb
33 root       20   0      0      0      0 S   0.0  0.0   0:00.00 cpuphp/4
34 root      -51   0      0      0      0 S   0.0  0.0   0:00.00 idle_inject/4
35 root       rt   0      0      0      0 S   0.0  0.0   0:00.76 migration/4
36 root       20   0      0      0      0 S   0.0  0.0   0:00.02 ksoftirqd/4
38 root       0 -20      0      0      0 I   0.0  0.0   0:00.00 kworker/4:0H-kb
39 root       20   0      0      0      0 S   0.0  0.0   0:00.00 cpuphp/5
40 root      -51   0      0      0      0 S   0.0  0.0   0:00.00 idle_inject/5
41 root       rt   0      0      0      0 S   0.0  0.0   0:00.76 migration/5
42 root       20   0      0      0      0 S   0.0  0.0   0:00.04 ksoftirqd/5
44 root       0 -20      0      0      0 I   0.0  0.0   0:00.00 kworker/5:0H-kb
45 root       20   0      0      0      0 S   0.0  0.0   0:00.00 cpuphp/6
46 root      -51   0      0      0      0 S   0.0  0.0   0:00.00 idle_inject/6
47 root       rt   0      0      0      0 S   0.0  0.0   0:00.76 migration/6
48 root       20   0      0      0      0 S   0.0  0.0   0:00.02 ksoftirqd/6
50 root       0 -20      0      0      0 I   0.0  0.0   0:00.00 kworker/6:0H-kb
51 root       20   0      0      0      0 S   0.0  0.0   0:00.00 cpuphp/7
52 root      -51   0      0      0      0 S   0.0  0.0   0:00.00 idle_inject/7
53 root       rt   0      0      0      0 S   0.0  0.0   0:00.76 migration/7
54 root       20   0      0      0      0 S   0.0  0.0   0:00.03 ksoftirqd/7
56 root       0 -20      0      0      0 I   0.0  0.0   0:00.00 kworker/7:0H-kb
57 root       20   0      0      0      0 S   0.0  0.0   0:00.00 kdevtmpfs
58 root       0 -20      0      0      0 I   0.0  0.0   0:00.00 netns
```

这时候我们可以通过如下命令查看指定进程的运行情况，例如：

查看kcalc进程的情况，命令：

```
top -p `pidof kcalc`

LinuxZn@LinuxZn: ~ 171x62
top - 01:34:46 up 2:41, 1 user, load average: 1.10, 0.80, 0.42
任务: 1 total, 0 running, 1 sleeping, 0 stopped, 0 zombie
%Cpu(s): 12.8 us, 1.3 sy, 0.0 ni, 85.6 id, 0.0 wa, 0.0 hi, 0.3 si, 0.0 st
KiB Mem : 8124336 total, 1997384 free, 3330976 used, 2795976 buff/cache
KiB Swap: 8787964 total, 8787964 free, 0 used, 4364084 avail Mem

进程 USER      PR  NI   VIRT   RES   SHR  %CPU %MEM    TIME+  COMMAND
5150 LinuxZn    20   0 901908 84412 67364 S   0.0  1.0   0:00.31 kcalc
```

这就简洁多了。

注意：

这里的”`号”并不是单引号！！

这里的”`号”并不是单引号！！

这里的”`号”并不是单引号！！

这个符号在键盘上感叹号!键的左边。

查看多个进程，如：

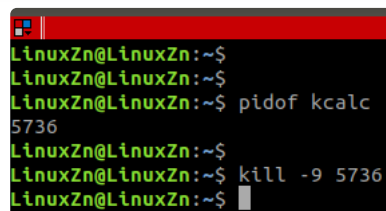
```
top -p `pidof kcalc` -p `pidof test_x86`
```

6、杀死进程

(1) 使用kill

先使用pidof查看进程的pid，然后再使用kill命令：

```
kill -9 process_pid
```

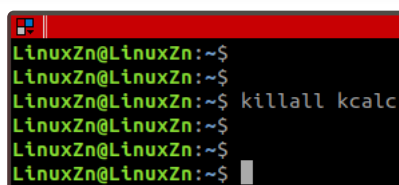
A terminal window with a red title bar. The prompt is 'LinuxZn@LinuxZn:~\$'. The user enters 'pidof kcalc' and the output is '5736'. Then the user enters 'kill -9 5736' and the prompt returns.

```
LinuxZn@LinuxZn:~$  
LinuxZn@LinuxZn:~$  
LinuxZn@LinuxZn:~$ pidof kcalc  
5736  
LinuxZn@LinuxZn:~$  
LinuxZn@LinuxZn:~$ kill -9 5736  
LinuxZn@LinuxZn:~$
```

(2) 使用killall

使用killall，命令：

```
killall process_name
```

A terminal window with a red title bar. The prompt is 'LinuxZn@LinuxZn:~\$'. The user enters 'killall kcalc' and the prompt returns.

```
LinuxZn@LinuxZn:~$  
LinuxZn@LinuxZn:~$  
LinuxZn@LinuxZn:~$ killall kcalc  
LinuxZn@LinuxZn:~$  
LinuxZn@LinuxZn:~$  
LinuxZn@LinuxZn:~$
```

7、终端输出的log同时保存到文件

有时候我们需要把终端实时输出的log信息保存到文件中，有如下两种方法。这三种方法也在之前的文章里写过，这里再简单提一下：

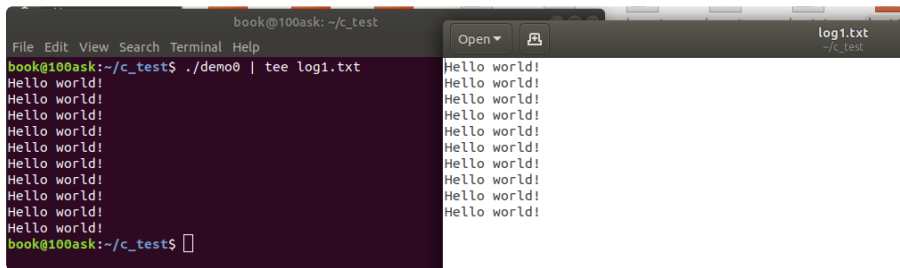
(1) tee

tee工具 用于将数据重定向到文件，另一方面还可以提供一份重定向数据的副本作为后续命令的stdin。简单的说就是 把数据重定向到给定文件和屏幕上。

命令：

```
executable_file | tee log_file
```

演示如下：



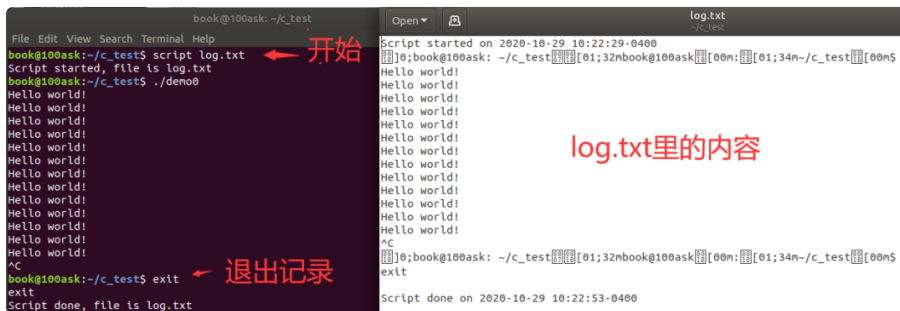
The image shows a terminal window and a separate window titled 'log1.txt'. The terminal window has a menu bar (File, Edit, View, Search, Terminal, Help) and shows the command 'book@100ask:~/c_test\$./demo0 | tee log1.txt'. It then prints 'Hello world!' ten times. The 'log1.txt' window shows the same ten 'Hello world!' lines.

(2) script

script工具 是一个非常使用的工具，可以把输出到终端的信息记录下来。使用步骤如：

- 输入 script log.txt 命令开始保存终端输出的信息，其中log.txt为需要写入的log文件，可随意命名。
- 输入 exit 退出保存。

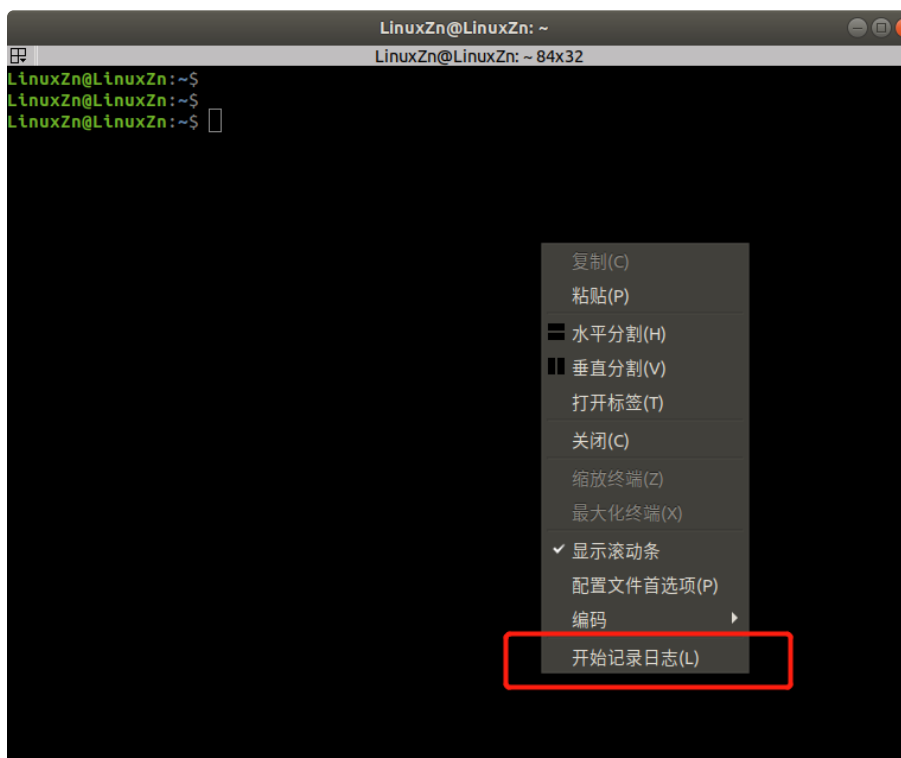
演示如下：



The image shows a terminal window and a separate window titled 'log.txt'. The terminal window has a menu bar (File, Edit, View, Search, Terminal, Help) and shows the command 'book@100ask:~/c_test\$ script log.txt'. It then prints 'Hello world!' ten times. The 'log.txt' window shows the output of the script, including the start time 'Script started on 2020-10-29 10:22:29-0400' and the ten 'Hello world!' lines. Red arrows and text are overlaid on the terminal window: '开始' (Start) points to the 'script log.txt' command, and '退出记录' (Exit record) points to the 'exit' command. The text 'log.txt里的内容' (Content in log.txt) is overlaid on the log.txt window.

(3) 使用一些可以保存log的终端工具

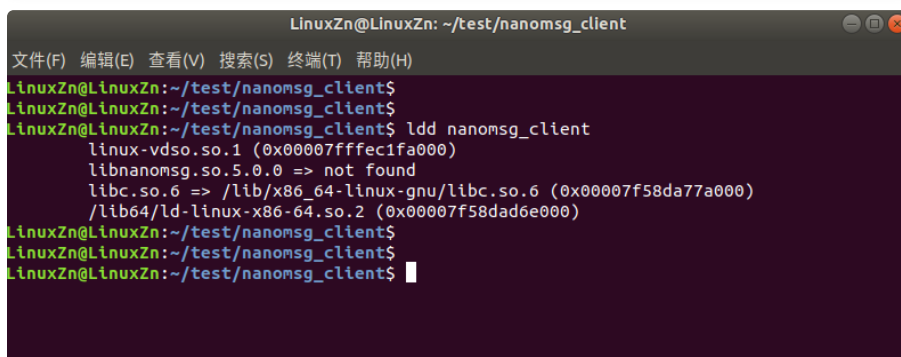
一些终端工具具有记录日志的功能，比如T Terminator终端：



8、查看程序依赖的动态库

有些程序依赖一些动态库，可以通过ldd命令查看依赖的动态库。命令：

```
ldd executable_file
```



9、查看ELF文件头

ELF文件有几种，可查看往期文章 [? ELF文件解析](#) 进行了解。之前刚来的一位应届生，编译了一份程序，编译没报错，但是一直运行不起来。然后在PC上运行有问题，报错如：

```
无法执行二进制文件：可执行文件格式错误
```

原因是他那份工程里设置了使用交叉编译器进行编译，但是他却在PC运行，所以就报错了。

我们可以查看可执行文件的ELF头，ELF头包含了很多信息，其中就包括有系统架构这一项。命令如：

```
readelf -h elf_file
```

```
LinuxZn@LinuxZn: ~/test
LinuxZn@LinuxZn: ~/test 84x32
LinuxZn@LinuxZn:~/test$
LinuxZn@LinuxZn:~/test$
LinuxZn@LinuxZn:~/test$ arm-linux-gnueabi-gcc test.c -o test_arm
LinuxZn@LinuxZn:~/test$
LinuxZn@LinuxZn:~/test$ readelf -h test_arm
ELF 头:
  Magic:      7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00
  类别:      ELF32
  数据:      2 补码, 小端序 (little endian)
  版本:      1 (current)
  OS/ABI:    UNIX - System V
  ABI 版本:  0
  类型:      EXEC (可执行文件)
  系统架构:  ARM
  版本:      0x1
  入口点地址: 0x102f1
  程序头起点: 52 (bytes into file)
  Start of section headers: 8512 (bytes into file)
  标志:      0x5000400, Version5 EABI, hard-float ABI
  本头的大小: 52 (字节)
  程序头大小: 32 (字节)
  Number of program headers: 8
  节头大小:  40 (字节)
  节头数量:  38
  字符串表索引节头: 35
LinuxZn@LinuxZn:~/test$
```

```
LinuxZn@LinuxZn: ~/test
LinuxZn@LinuxZn: ~/test 84x32
LinuxZn@LinuxZn:~/test$
LinuxZn@LinuxZn:~/test$ gcc test.c -o test_x86
LinuxZn@LinuxZn:~/test$
LinuxZn@LinuxZn:~/test$ readelf -h test_x86
ELF 头:
  Magic:      7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00
  类别:      ELF64
  数据:      2 补码, 小端序 (little endian)
  版本:      1 (current)
  OS/ABI:    UNIX - System V
  ABI 版本:  0
  类型:      DYN (共享目标文件)
  系统架构:  Advanced Micro Devices X86-64
  版本:      0x1
  入口点地址: 0x530
  程序头起点: 64 (bytes into file)
  Start of section headers: 6440 (bytes into file)
  标志:      0x0
  本头的大小: 64 (字节)
  程序头大小: 56 (字节)
  Number of program headers: 9
  节头大小:  64 (字节)
  节头数量:  29
  字符串表索引节头: 28
LinuxZn@LinuxZn:~/test$
```

除此之外，通过file命令也可以查看到文件的一些信息：

```
LinuxZn@LinuxZn: ~/test
LinuxZn@LinuxZn: ~/test 93x9
LinuxZn@LinuxZn:~/test$
LinuxZn@LinuxZn:~/test$ file test_arm
test_arm: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-armhf.so.3, for GNU/Linux 2.6.32, BuildID[sha1]=80c21cfe197a86f7d3f736e27b31c9bfe4a6e004, with debug_info, not stripped
LinuxZn@LinuxZn:~/test$
```

10、文本文件查看

文本文件查看我们一般使用cat命令，但除了cat命令之外，还有其它几个实用的命令，下

面依次来介绍：

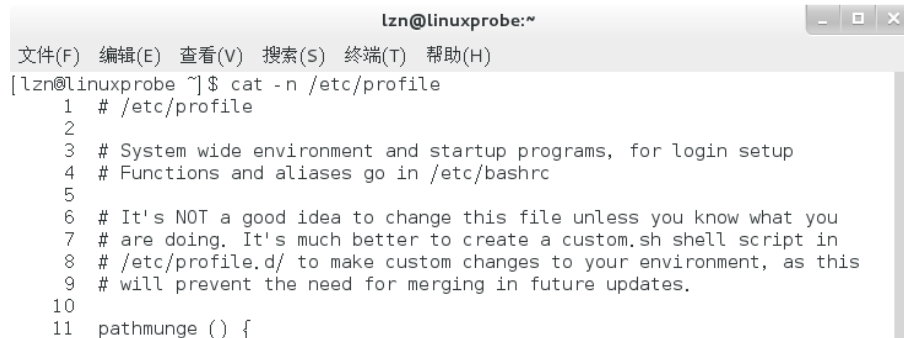
(1) cat命令

cat命令常用于查看内容较少的文件。很多人把这个命令叫做 小猫咪 命令，但cat其实是 concatenate（连续）的缩写，即连续显示文本内容。命令格式为：

```
cat [参数选项] [文件]
```

如：

```
cat -n /etc/profile
```



```
lzn@linuxprobe:~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
[lzn@linuxprobe ~]$ cat -n /etc/profile  
1  # /etc/profile  
2  
3  # System wide environment and startup programs, for login setup  
4  # Functions and aliases go in /etc/bashrc  
5  
6  # It's NOT a good idea to change this file unless you know what you  
7  # are doing. It's much better to create a custom.sh shell script in  
8  # /etc/profile.d/ to make custom changes to your environment, as this  
9  # will prevent the need for merging in future updates.  
10  
11 pathmunge () {
```

其中，加上参数 -n 用可以显示行数。cat的更多的参数选项可以输入 man cat 进行查看。以下列举的其他命令的详细介绍也可以输入 man 命令 进行查看。

(2) tac命令

tac命令的正好与cat命令相反，是从文件末尾开始显示。

(3) more命令

more命令适用于查看内容较多的文件。因为他可以实时显示百分比以提示现在已经阅读了多少内容。

命令格式为：

```
more [参数选项] [文件]
```

如：

```
lzn@linuxprobe:~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
# /etc/profile

# System wide environment and startup programs, for login setup
# Functions and aliases go in /etc/bashrc

# It's NOT a good idea to change this file unless you know what you
# are doing. It's much better to create a custom.sh shell script in
# /etc/profile.d/ to make custom changes to your environment, as this
# will prevent the need for merging in future updates.

pathmunge () {
    case "${PATH}:" in
        *:"$1":*)
            ;;
        *)
            if [ "$2" = "after" ] ; then
                PATH=$PATH:$1
            else
                PATH=$1:$PATH
            fi
    esac
}

-- More -- (35%)
```

可以使用 空格键 或者 回车键 往下翻页查看后面的内容。

(4) less命令

less也适用于查看内容较多的文件。less比more更为灵活，因为less可以往上、往下翻页。按下键盘上 PgUp 键可以往上翻页，按下 PgDn 可以往下翻页。但是less命令不会实时显示当前阅读的百分比。

命令格式为：

```
less [参数选项] [文件]
```

(5) head命令

head命令用于查看文件的前n行。如使用命令

```
head -n 20 /etc/profile
```

查看/etc目录下profile文件的前20行内容：

```
lzn@linuxprobe:~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
[lzn@linuxprobe ~]$ head -n 20 /etc/profile
# /etc/profile

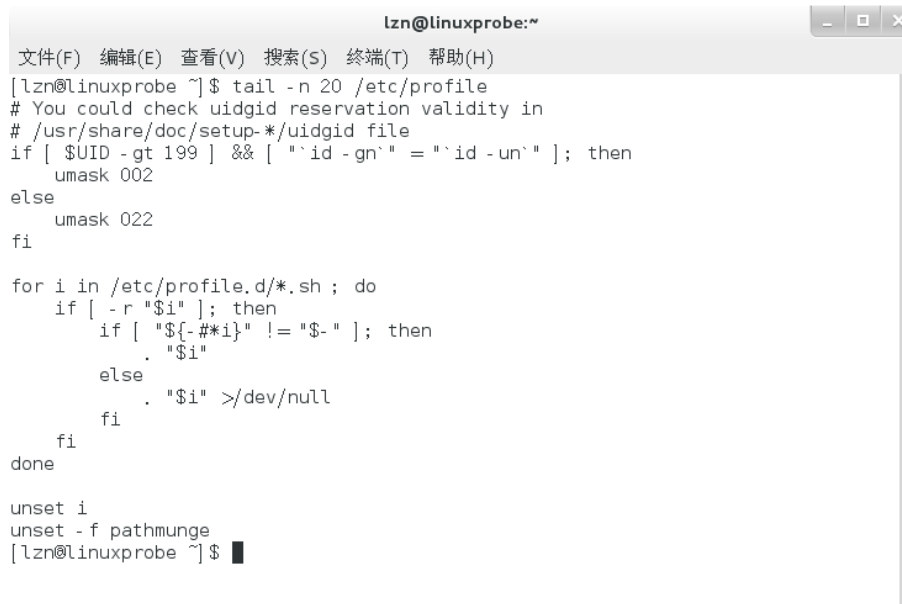
# System wide environment and startup programs, for login setup
# Functions and aliases go in /etc/bashrc

# It's NOT a good idea to change this file unless you know what you
# are doing. It's much better to create a custom.sh shell script in
# /etc/profile.d/ to make custom changes to your environment, as this
# will prevent the need for merging in future updates.

pathmunge () {
    case "${PATH}:" in
        *:"$1":*)
            ;;
        *)
            if [ "$2" = "after" ] ; then
                PATH=$PATH:$1
            else
                PATH=$1:$PATH
            fi
    esac
}
[lzn@linuxprobe ~]$
```

(6) tail命令

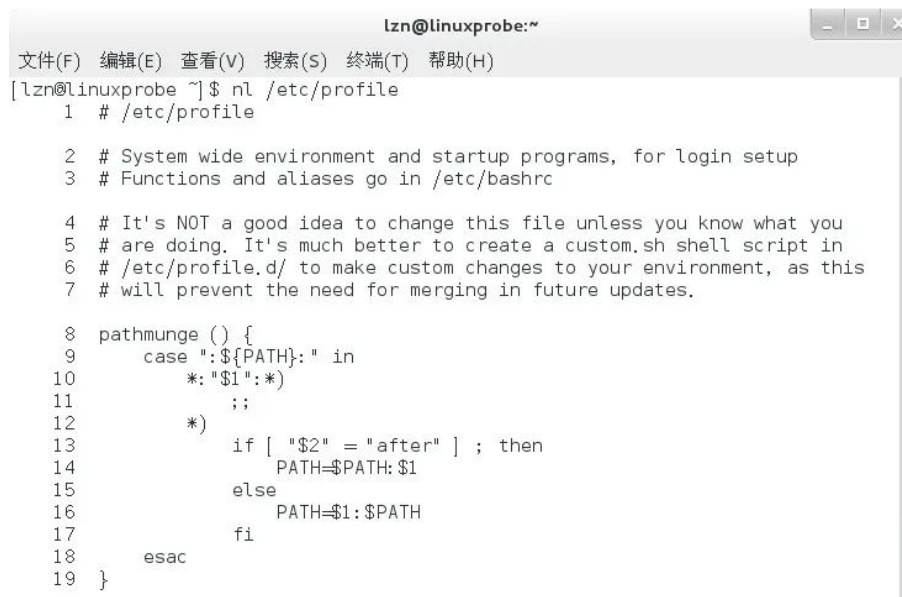
tail命令与head命令相反，tail命令用于查看文件后n行内容。如：



```
lzn@linuxprobe:~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
[lzn@linuxprobe ~]$ tail -n 20 /etc/profile  
# You could check uidgid reservation validity in  
# /usr/share/doc/setup-*/uidgid file  
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then  
    umask 002  
else  
    umask 022  
fi  
  
for i in /etc/profile.d/*.sh ; do  
    if [ -r "$i" ]; then  
        if [ "${-##i}" != "$-" ]; then  
            . "$i"  
        else  
            . "$i" >/dev/null  
        fi  
    fi  
done  
  
unset i  
unset -f pathmunge  
[lzn@linuxprobe ~]$
```

(7) nl命令

nl命令可以显示内容的同时显示行号，与 `cat -n` 命令的作用差不多：



```
lzn@linuxprobe:~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
[lzn@linuxprobe ~]$ nl /etc/profile  
1  # /etc/profile  
  
2  # System wide environment and startup programs, for login setup  
3  # Functions and aliases go in /etc/bashrc  
  
4  # It's NOT a good idea to change this file unless you know what you  
5  # are doing. It's much better to create a custom.sh shell script in  
6  # /etc/profile.d/ to make custom changes to your environment, as this  
7  # will prevent the need for merging in future updates.  
  
8  pathmunge () {  
9      case "${PATH}:" in  
10         *: "$1":*)  
11             ;;  
12         *)  
13             if [ "$2" = "after" ] ; then  
14                 PATH=$PATH:$1  
15             else  
16                 PATH=$1:$PATH  
17             fi  
18         esac  
19     }
```

11、设置LD_LIBRARY_PATH

LD_LIBRARY_PATH 是Linux / Unix中预定义的环境变量，它设置链接器在链接动态库/共享库时应该查看的路径。有时候需要把当前路径加到LD_LIBRARY_PATH中，如：

```
export LD_LIBRARY_PATH=./:$LD_LIBRARY_PATH
```

以上就是本次分享的一些实用的小技巧。

往期好文

[嵌入式设备AP配网实例分享](#)

[嵌入式Linux单板连接飞燕物联网平台](#)

[分享一种灵活性很高的协议格式（附代码例子）](#)

[嵌入式大杂烩周记 | 第 16 期](#)

[嵌入式大杂烩周记 | 第 15 期](#)

[访问非法内存为什么不会出错？](#)

[嵌入式大杂烩周记 | 第 14 期](#)

[分享几个实用的代码片段（第二弹）](#)

[分享一种你可能不知道的bug定位方法](#)

[分享一种修改配置文件的方法](#)

[《嵌入式大杂烩周记第 13 期：Iz4》](#)

[《嵌入式并行多线程处理器，了解一下！》](#)

[《分享一种修改配置文件的方法》](#)

[《分享几个实用的代码片段（附代码例子）》](#)

[《废旧板子再利用：搭建无线调试环境！》](#)

[《嵌入式段错误的3种调试方法汇总！》](#)

[《简说TCP通信非阻塞接收（附代码例子）》](#)

[《TCP通信用接口的使用封装》](#)

[《嵌入式软件中，总线错误的坑？替大家先踩一步》](#)

[《分享嵌入式软件调试方法及几个有用的工具！》](#)

[《分享两点提高编程能力的建议！》](#)



嵌入式大杂烩

本公众号专注于嵌入式技术，包括但不限于C/C++、嵌入式、物联网、Linux等编程学...
304篇原创内容

公众号

在公众号聊天界面回复**1024**，可获取嵌入式资源；回复 **m** ，可查看文章汇总

[阅读原文](#)

喜欢此内容的人还喜欢

这5个Linux安全相关的命令，很实用！

网络技术联盟站



2万字总结，体系化带你全面认识 Nginx ！

Java知音



数据库联合查询性能到底如何？

代码小铺



